**ONLINE STORE APPLICATION PROGRAMMING INTERFACE (API) DOCUMENTATION**

**INTRODUCTION**

This document is an API integration document for an online store API Interface. It provides guidance and template material which is intended to assist the relevant stakeholders to easily integrate to the different interface.

The API interface uses JSON over HTTPS to process incoming requests, sample requests are provided further in this document.

**ASSUMPTIONS**

In order for the relevant stakeholders to integrate via the API Interface, the assumption is that such an individual has;

* Familiarity with the TCP/IP stack.
* Familiarity with Transport Layer Security schemes.
* Familiarity with SOA and RESTful services.
* Familiarity with JSON.

**INTEGRATION OVERVIEW**

The API Interface is a secure interface and has the capabilities to do the following:

|  |  |
| --- | --- |
| **API Listing** | |
| Fetch Inventory | Get inventory items information |
| Fetch Supplier | Get supplier information |
| Add Inventory Item | Add inventory item to the list of the inventory |
| Add Supplier | Add supplier information to list of suppliers |
| Update Inventory Item | Update inventory item information |
| Update Supplier | Update suppliers’ information |
| Delete Inventory Item | Delete inventory item |
| Delete Supplier | Delete supplier |

**DEPLOYMENT OVERVIEW**

**Requirements:**

1. Compressed website files (tested and working on localhost). Zip format  
   recommended, which will be sent alongside this documentation.
2. requirements.txt (a list of the Django packages for your application), which is embedded in the Zip file above.

**Procedure:**

1.Login to your Cpanel; Under Software, select ‘Setup python App’

2. Click on ‘Create Application’

3. In the create Application Page:

1. Python Version: Select the python version
2. Application root: Set the root application folder for your application. It is advisable to use the domain name.
3. Application Url: select the domain on which you wish to install the Django application on.
4. Leave the other fields blank; they will be pre-filled once the application has been created; then click on ‘CREATE’

4. Copy the highlighted text below after creation which is the env environment that you be working with for use in a later stage

5. Navigate to the root folder of your project and upload the compressed project

folder; extract the files.

6. Set the database credentials in the .env file in the project folder

7. On your root directory; select and edit passenger\_wsgi.py; delete the content and

add the following line;

from online\_store\_api.wsgi import application

where online\_store\_api is the folder containing the wsgi file.

8. On your cpanel homepage, select terminal under Advanced group and do the following;

1. pip install -r requirements.txt → to install all the modules for our project
2. python manage.py makemigrations → To create project tables
3. python manage.py migrate → to map project tables to our database
4. python manage.py createsuperuser → To create a super user /administrator account

9. The Api should be up and running. In case you have a different use case please reach out for assistance.

**API USAGE**

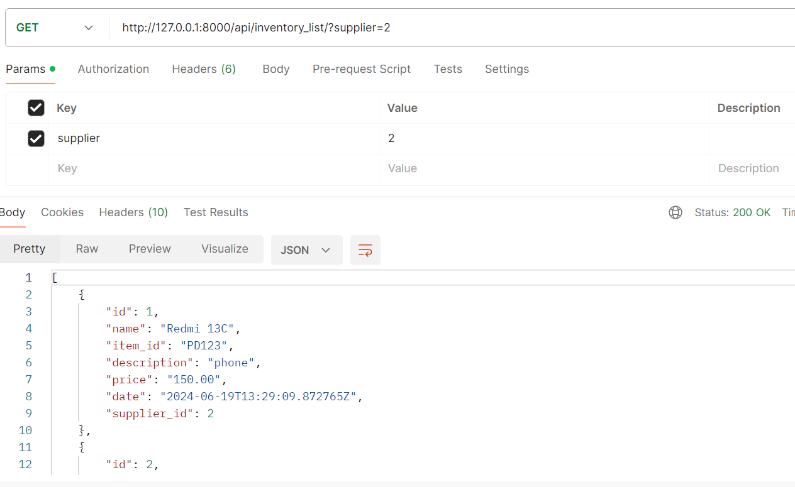
**Fetch Inventory**

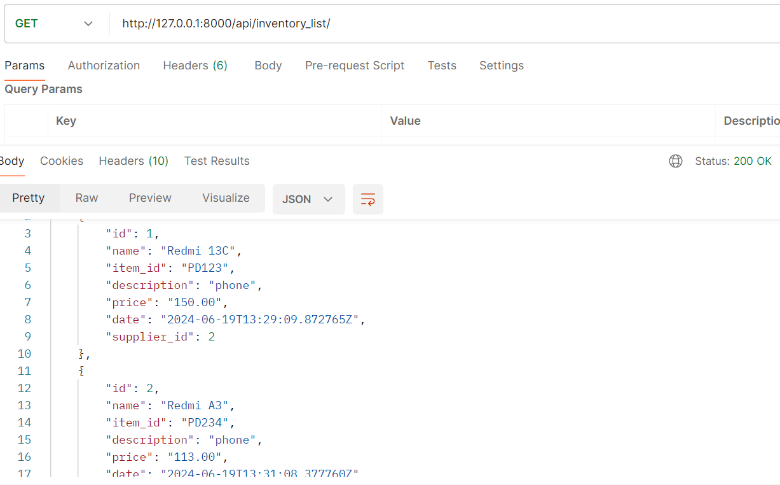
This endpoint is used by a client to fetch information pertaining to the Inventory. It can also be used to fetch items associated with a supplier by passing the suppliers ID which gotten from the API associated with supplier. Items can also be fetched by appending their name.

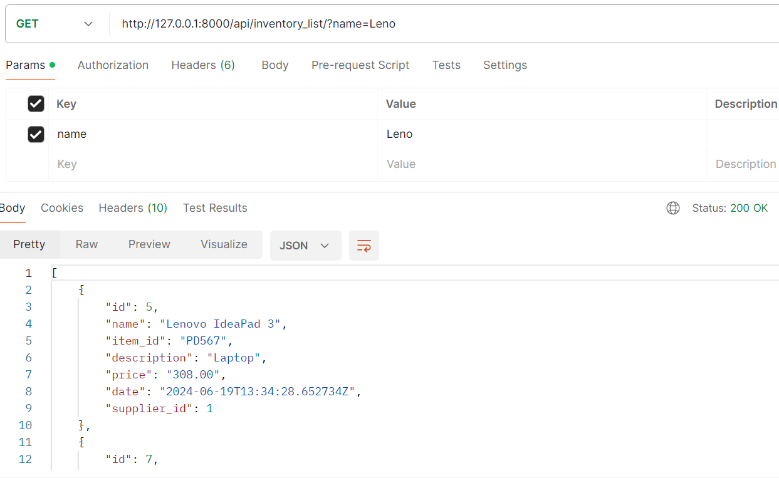
API structure can be examined in the table below;

|  |  |
| --- | --- |
| URL | http://127.0.0.1:8000/api/inventory\_list/ |
| Method | GET |
| Content/Type | text/plain |
| JSON Object | *Not Applicable* |
| Request Object | *Not Applicable* |

Screenshot example associated with API call;







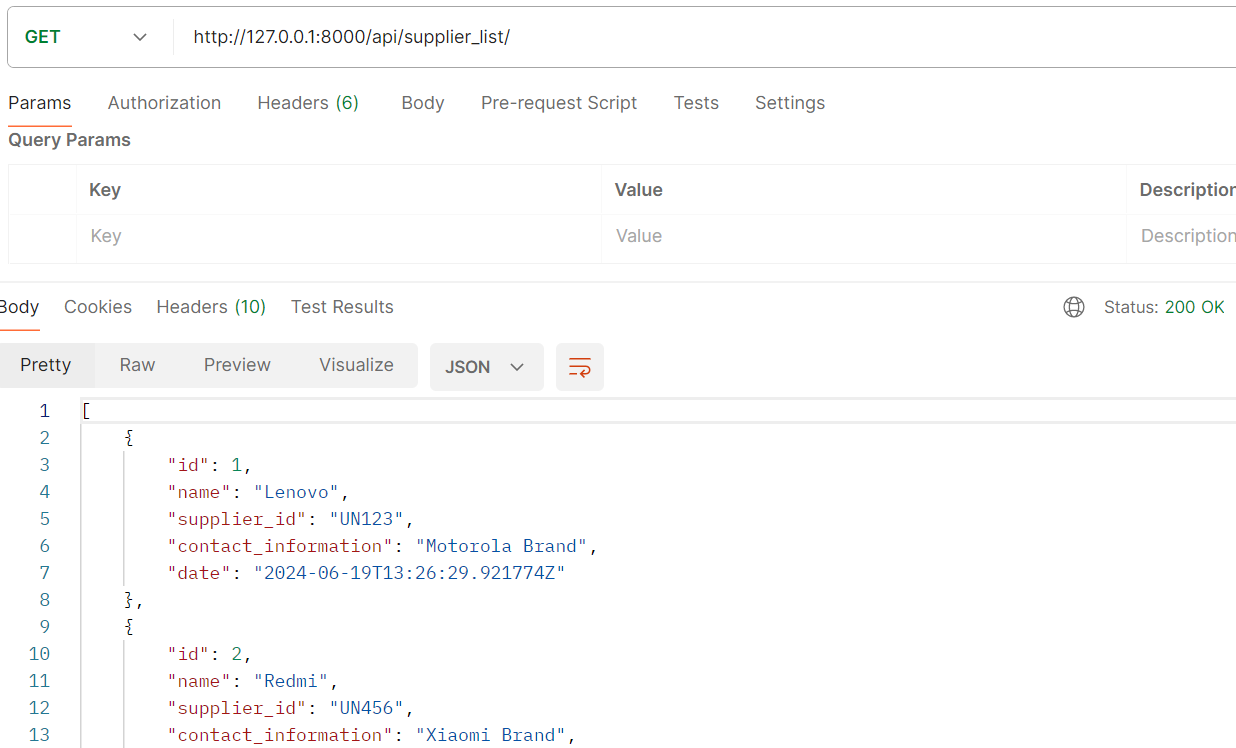
**Fetch Supplier**

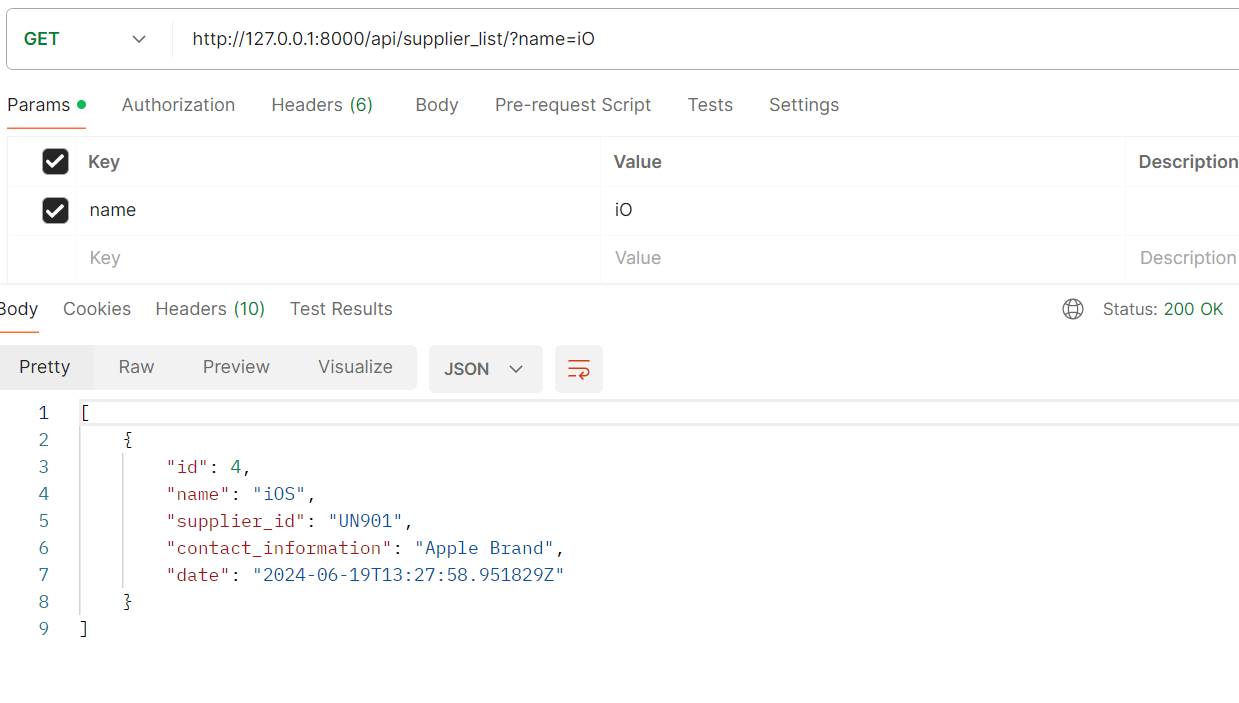
This endpoint is used by a client to fetch information pertaining to the Supplier’s information. Supplier can also be fetched by appending their name.

API structure can be examined in the table below;

|  |  |
| --- | --- |
| URL | http://127.0.0.1:8000/api/supplier\_list/ |
| Method | GET |
| Content/Type | text/plain |
| JSON Object | *Not Applicable* |
| Request Object | *Not Applicable* |

Screenshot example associated with API call;





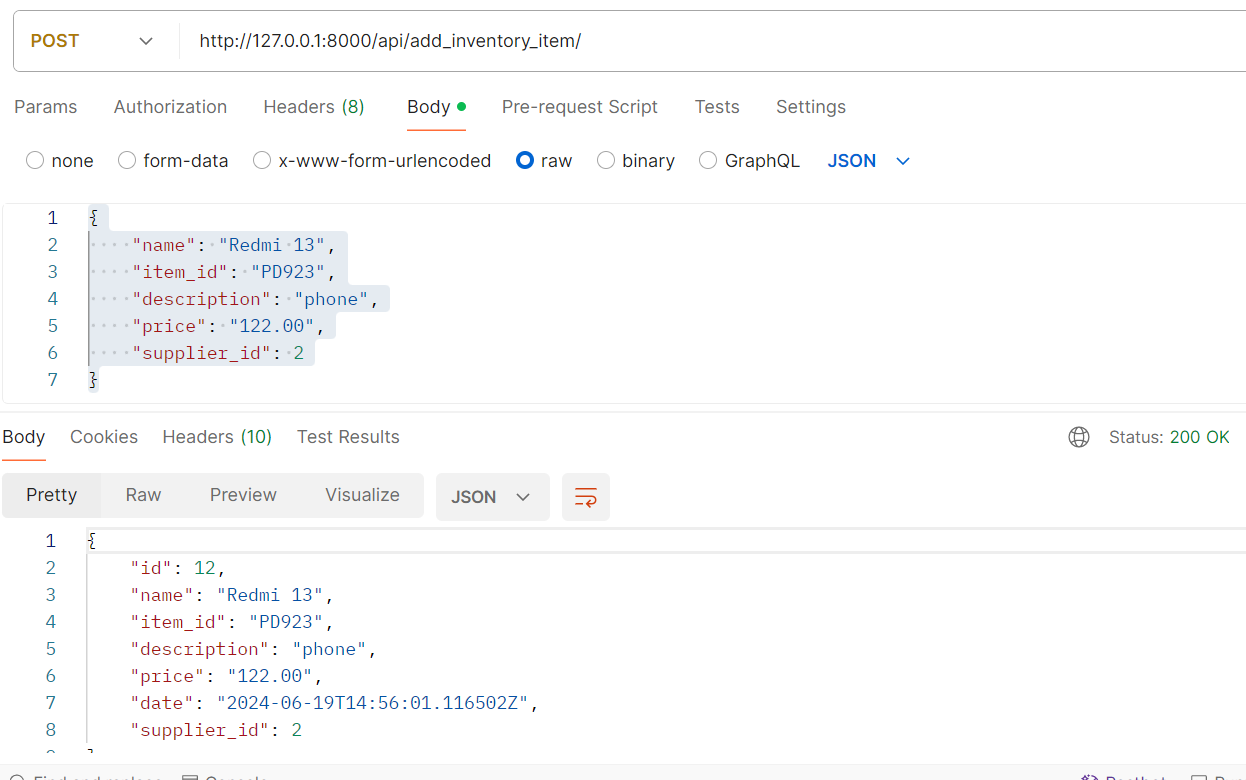
**Add Inventory Item**

This endpoint is used by a client to add an item to the Inventory.

API structure can be examined in the table below;

|  |  |
| --- | --- |
| URL | http://127.0.0.1:8000/api/add\_inventory\_item/ |
| Method | POST |
| Content/Type | application/json |
| JSON Object | {  "name": "string",  "item\_id": " string ",  "description": "string”,  "price": " string ",  "supplier\_id": interger  } |
| Request Object | JSON Object |

Screenshot example associated with API call;



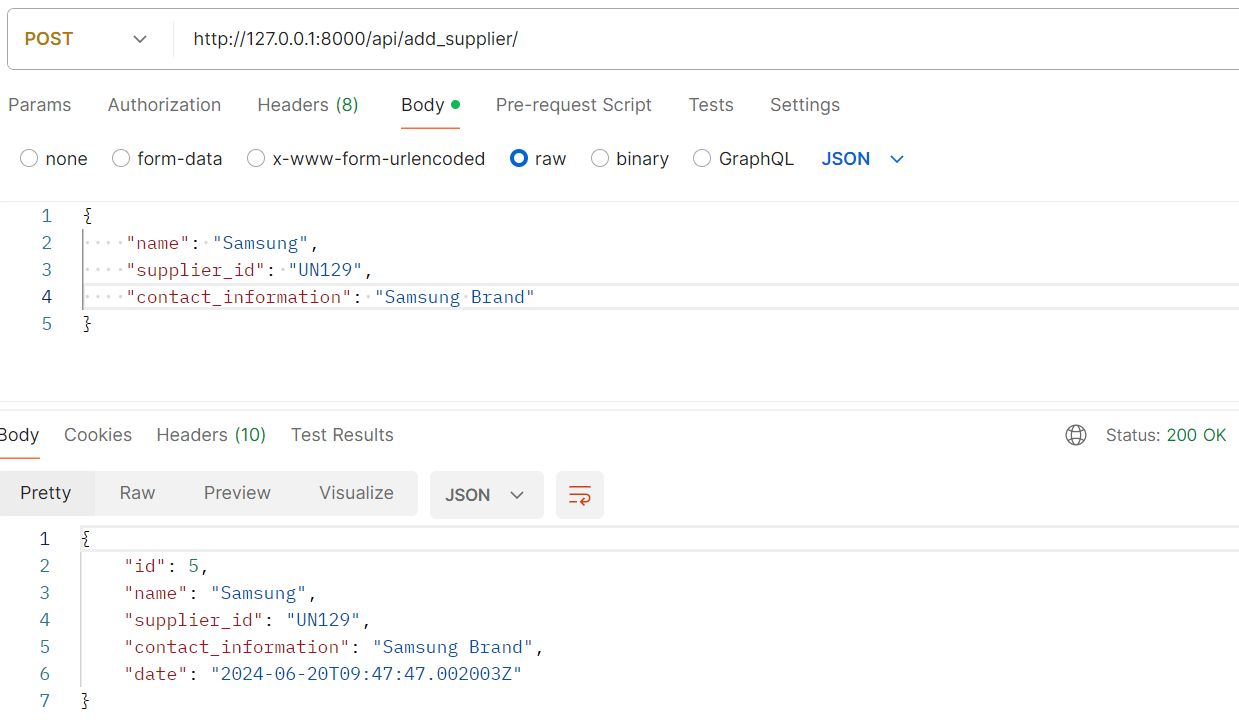
**Add Supplier**

This endpoint is used by a client to add a supplier.

API structure can be examined in the table below;

|  |  |
| --- | --- |
| URL | http://127.0.0.1:8000/api/add\_inventory\_item/ |
| Method | POST |
| Content/Type | application/json |
| JSON Object | {  "name": "string",  "supplier\_id": "string",  "contact\_information": "string"  } |
| Request Object | JSON Object |

Screenshot example associated with API call;



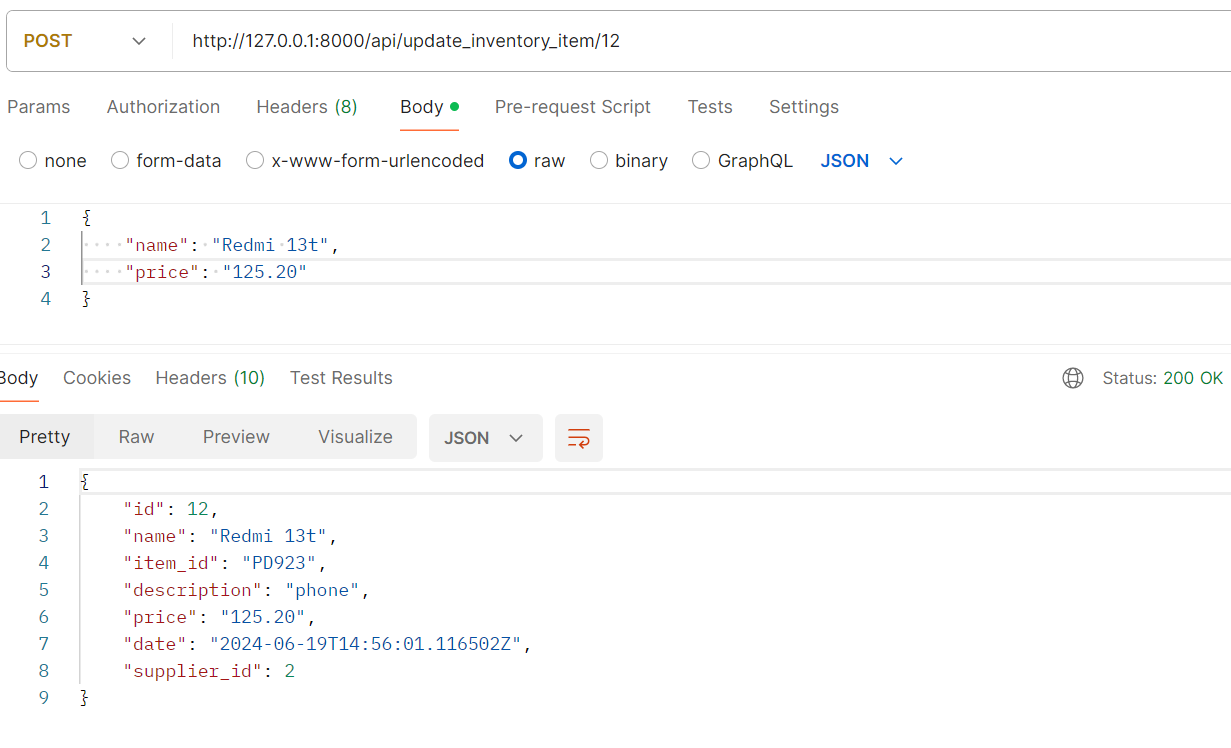
**Update Inventory Item**

This endpoint is used by a client to update an item in the Inventory. This is done by append the ‘id’ of item to the endpoint and passing the JSON payload to the endpoint.

API structure can be examined in the table below;

|  |  |
| --- | --- |
| URL | http://127.0.0.1:8000/api/update\_inventory\_item/ |
| Method | POST |
| Content/Type | application/json |
| JSON Object | {  "name": "string",  "item\_id": "string",  "description": "string",  "price": "string",  "supplier\_id": interger  } |
| Request Object | JSON Object |

Screenshot example associated with API call;



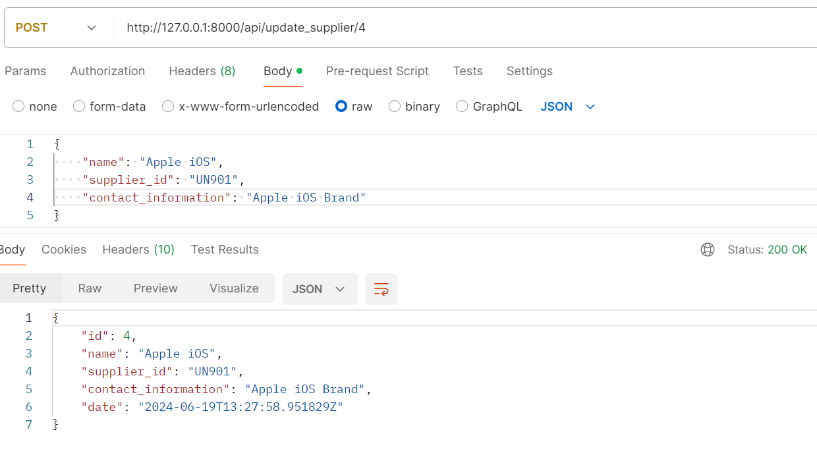
**Update Supplier**

This endpoint is used by a client to update a supplier. This is done by append the ‘id’ of supplier to the endpoint and passing the JSON payload to the endpoint.

API structure can be examined in the table below;

|  |  |
| --- | --- |
| URL | http://127.0.0.1:8000/api/update\_inventory\_item/ |
| Method | POST |
| Content/Type | application/json |
| JSON Object | {      "name": "string",      "supplier\_id": "string",      "contact\_information": "string"  } |
| Request Object | JSON Object |

Screenshot example associated with API call;



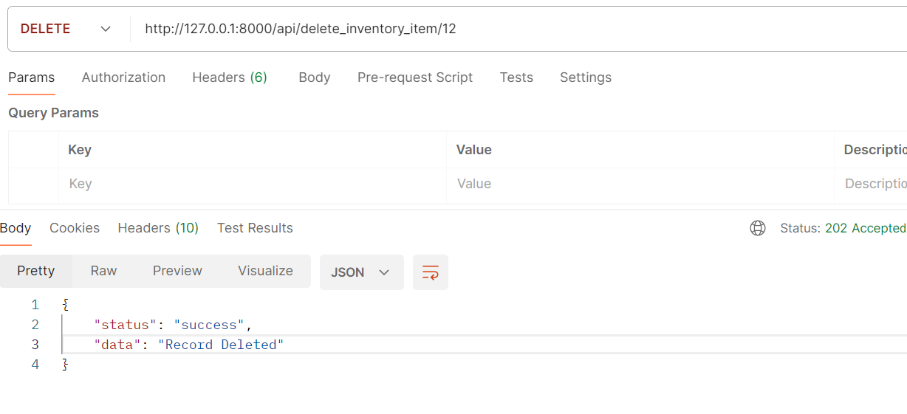
**Delete Inventory Item**

This endpoint is used by a client to delete an item in the Inventory. This is done by append the ‘id’ of item to the endpoint.

API structure can be examined in the table below;

|  |  |
| --- | --- |
| URL | http://127.0.0.1:8000/api/delete\_inventory\_item/ |
| Method | DELETE |
| Content/Type | application/json |
| JSON Object | *Not Applicable* |
| Request Object | *Not Applicable* |

Screenshot example associated with API call;



**Delete Supplier**

This endpoint is used by a client to delete a supplier. This is done by append the ‘id’ of supplier to the endpoint.

API structure can be examined in the table below;

|  |  |
| --- | --- |
| URL | http://127.0.0.1:8000/api/delete\_inventory\_item/ |
| Method | DELETE |
| Content/Type | application/json |
| JSON Object | *Not Applicable* |
| Request Object | *Not Applicable* |

Screenshot example associated with API call;

