PIZZA APP

Database

Table Name	Fields	Data Type	Max Length	Entered By
Pizza Specification	Pizza Size	String	30	Randomly Generated
	Pizza Toppings	List of Strings	30	User
Pizza Instantiation	Pizza Id	Integer	Not Specified	Auto
	Pizza Type	String	20	User
	Pizza Size	String	30	User
	Pizza Toppings	String	30	User

There are 2 tables.

1. **Pizza Specification**

It stores the already defined specifications by the Owner. Its fields can be changed.

It has 2 fields.

1. Pizza Size which is specified in advance. It can be Small, Medium, Large etc.
2. Pizza Toppings which is a List of Toppings. It can be Onion, Tomato, Corn, Capsicum, Cheese, Jalapeno etc.

2. **Pizza Instantiation**

It stores 1 Pizza at a time. Pizza Product Id is generated by default. Pizza Type is randomly generated. Pizza Size and Toppings are mentioned but added only when they are already specified in in Pizza Specification Table.

It has 4 fields.

1. Pizza Id which is auto-generated with every new entry.
2. Pizza Type which is randomly generated. It takes two values: Regular and Square.
3. Pizza Size which is specified by user while creating the pizza product. The entered size is matched against the already specified List of Pizza Size in Pizza Specification Table.
4. Pizza Toppings which is specified by user while creating the pizza product. The entered topping is matched against the already specified List of Pizza Toppings in Pizza Specification Table.

API

The API has two classes.

1. **PizzaSpecificationView**

Endpoint: pizza_specification

Methods: POST and GET

1. **POST**

The Owner defines the specification for Pizza Size and Pizza Toppings using POST method.

Query

http://127.0.0.1:8000/pizza_specification

```
{
    "pizza_size ": ["small", "large"],
    "pizza_toppings": ["cheese", "corn"]
}
```

Result

For Success -> "success": "Data saved successfully"

Or

For Failure -> "error": "Data couldn't be saved"

2. **GET**

The Owner and others can see the specification for Pizza Size and Pizza Toppings using GET method.

Query

http://127.0.0.1:8000/pizza_specification

Result

For Success

```
{
    "pizza_size ": ["small", "large"],
    "pizza_toppings": ["cheese", "corn"]
}
```

For Failure

"error": "Data couldn't be found"

Endpoints: pizza_instantiation and pizza_instantiation/search_by

Methods: POST, GET, PUT, and DELETE

1. **POST**

The Owner creates a Pizza product using POST method.

The Pizza size and Pizza topping entered by him/her are first checked in the specification table. If they exist, a new product is allowed to be created.

Query

http://127.0.0.1:8000/pizza_instantiation

```
{
    "pizza_size ": "large",
    "pizza_toppings": "cheese"
}
```

Pizza Id and Pizza Type are automatically randomly entered.

Result

For Success -> "success": "Data saved successfully"

Or

For Failure -> "error": "Data couldn't be saved"

2. GET

The attributes of the Pizza can be seen using GET method.

We can search:

i. all the products

ii. by size

iii. by toppings

Query 1

http://127.0.0.1:8000/pizza_ instantiation

Result

For Success

```
{
    "pizza_id": 1,
    "pizza_type": "regular",
    "pizza_size": "large",
    "pizza_toppings": "cheese"
}
```

For Failure

"error": "Data couldn't be found"

Query 2

http://127.0.0.1:8000/pizza_ instantiation/search_by_type/large

Result

For Success

{

"pizza_id": 1,

"pizza_type": "regular",

"pizza_size": "large",

"pizza_toppings": "cheese"

}

For Failure

"error": "Data couldn't be found"

Query 3

http://127.0.0.1:8000/pizza_ instantiation/search_by_type/cheese

Result

For Success

{

"pizza_id": 1,

"pizza_type": "regular",

"pizza_size": "large",

"pizza_toppings": "cheese"

}

For Failure

"error": "Data couldn't be found"

3. PUT

The Owner updates a Pizza product using PUT method.

The Pizza size and Pizza topping entered by him/her are first checked in the specification table. If they exist, a new product is allowed to be updated.

Query

http://127.0.0.1:8000/pizza_instantiation/1

{

"pizza_size ": "large",

"pizza_toppings": "corn"

}

Result

For Success -> "success": "Data updated successfully"

Or

For Failure -> "error": "Data couldn't be updated"

4. DELETE

The Owner deletes a Pizza product using DELETE method, either a single product using pizza_id or all the products.

Query

http://127.0.0.1:8000/pizza_instantiation/1

Result

For Success -> "success": "Data deleted successfully"

Or

For Failure -> "error": "Data couldn't be deleted"