**Restful APi for Project Management**

**(Documentation)**

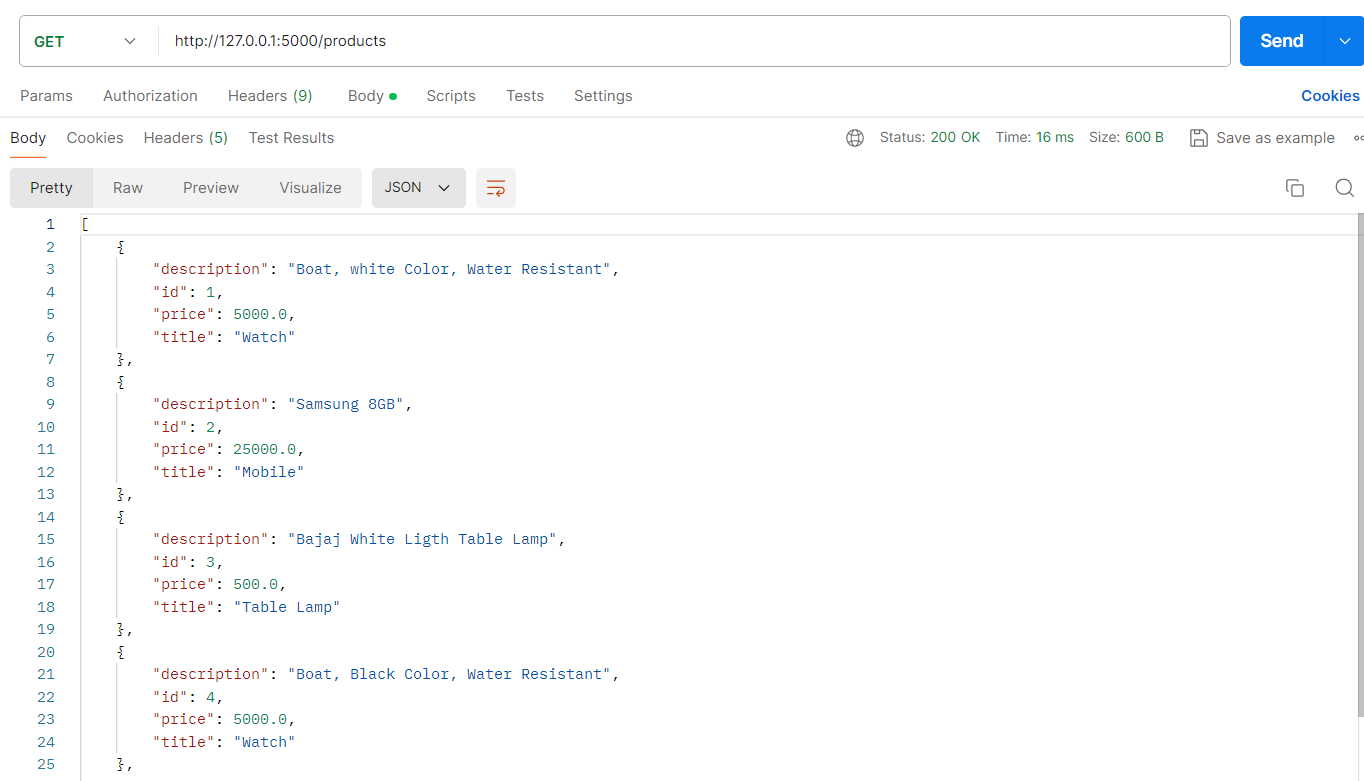
**1.Introduction :**

The api is developed using a python webframework that is Flask and we have used sqllite database.Here sqllite provides the restful interface for interacting with the data. It follows the principle of Representation state transfer allowing the user to perform CRUD operations using the HTTP methods.The API provides endpoints for accessing and manipulating data stored in the SQLite database, making it easy for clients to integrate with and manage data remotely.

**2.Example Request:**

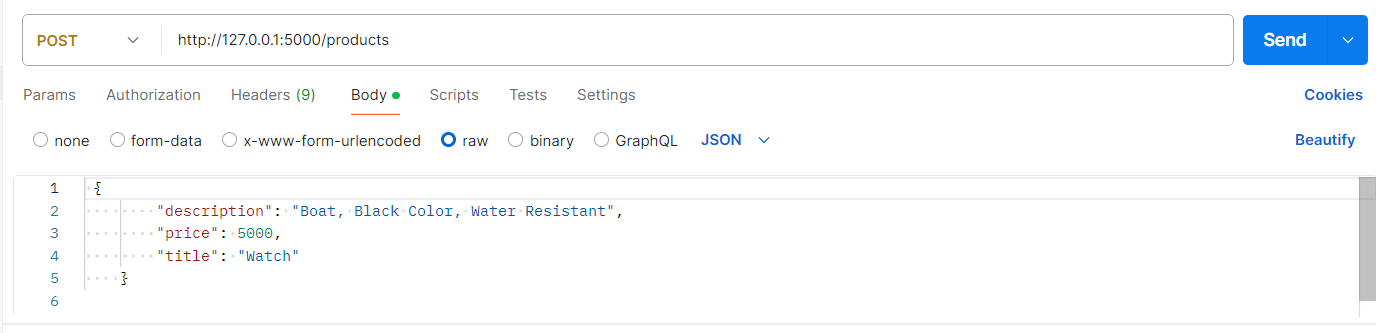
1.GET:

URL : <http://127.0.0.1:5000/products>



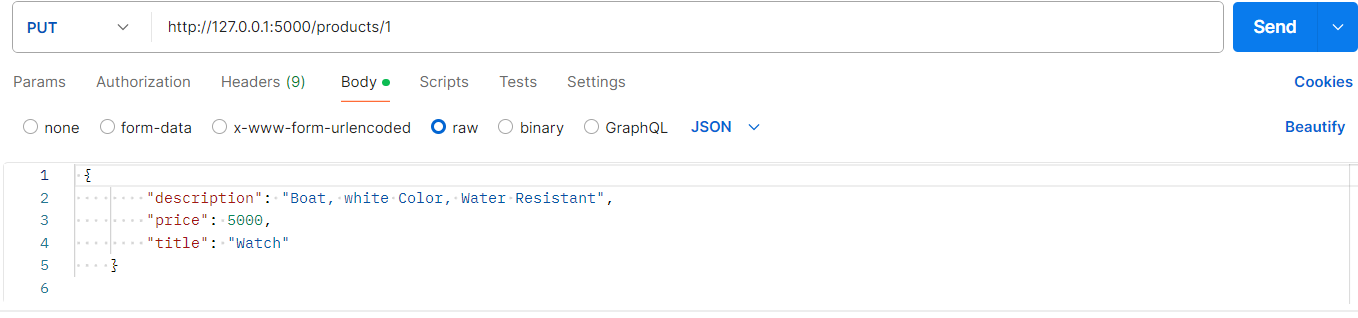
2.POST:

URL: <http://127.0.0.1:5000/products>



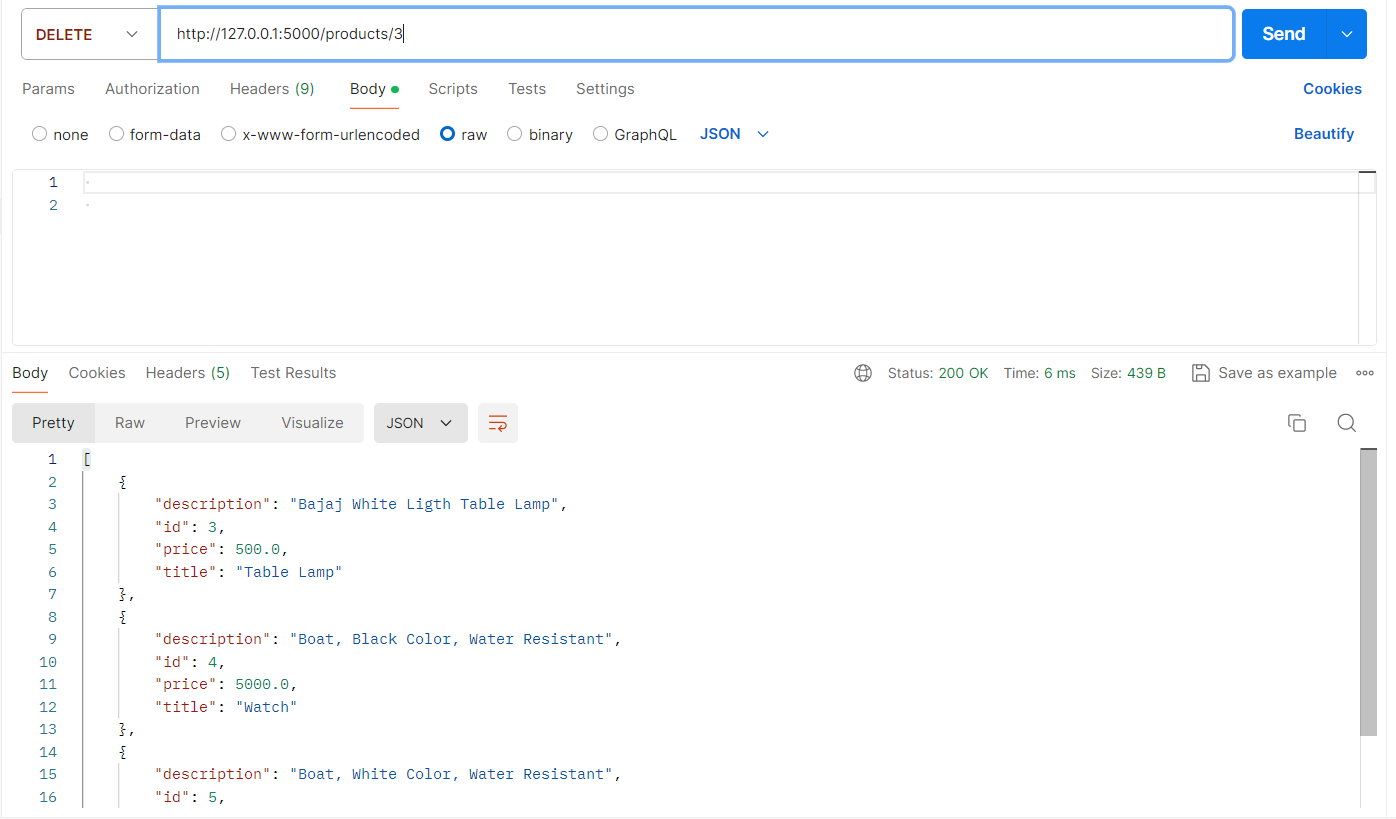
3.PUT:

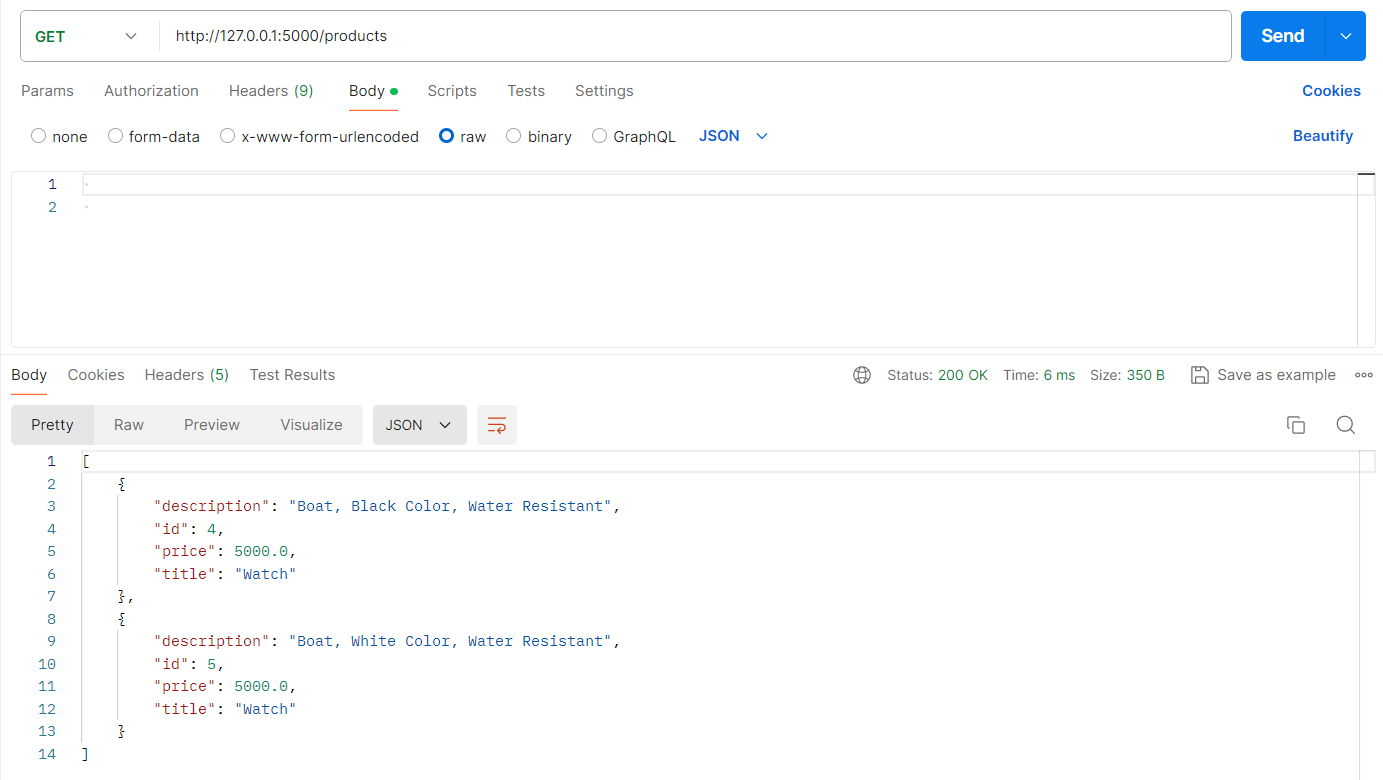
URL: <http://127.0.0.1:5000/products>/1



4.DELETE:

URL: <http://127.0.0.1:5000/products>/1





**3.Code Samples:**

1.GET:

@app.route('/products',methods = ['GET'])

def get\_products():

products = Product.query.all()

products\_list=[]

for product in products:

product\_dict={

'id': product.id,

'title': product.title,

'description': product.description,

'price': product.price

}

products\_list.append(product\_dict)

return jsonify(products\_list)

2.GET BY ID:

@app.route('/products/<int:id>',methods = ['GET'])

def get\_product(id):

product=Product.query.get\_or\_404(id)

product\_Dict={

'id': product.id,

'title':product.title,

'description':product.description,

'price':product.price

}

return jsonify(product\_Dict)

3.POST:

@app.route('/products',methods = ['POST'])

def create\_product():

try:

data = request.get\_json()

product = Product(title=data['title'], description=data['description'], price=data['price'])

db.session.add(product)

db.session.commit()

Product\_details = {

'id': product.id,

'title': product.title,

'description': product.description,

'price': product.price

}

response = jsonify(Product\_details)

response.status\_code = 201

return response

except KeyError as e:

return jsonify({'error': f'Missing key: {e}'}), 400

except Exception as e:

return jsonify({'error': str(e)}), 400

4.PUT:

@app.route('/products/<int:id>', methods = ['PUT'])

def update\_product(id):

try:

data = request.get\_json()

product = Product.query.get\_or\_404(id)

product.title = data['title']

product.description = data['description']

product.price = data['price']

db.session.commit()

return jsonify({

'id': product.id,

'title': product.title,

'description': product.description,

'price': product.price

})

except KeyError as e:

return jsonify({'error': f'Missing key: {e}'}), 400

except Exception as e:

return jsonify({'error': str(e)}), 400

5.DELETE:

@app.route('/products/<int:id>', methods = ['DELETE'])

def delete\_product(id):

prod = Product.query.get\_or\_404(id)

db.session.delete(prod)

db.session.commit()

return '',204

ERROR HANDLER:

@app.errorhandler(404)

def not\_found(error):

return jsonify({'error': 'Not Found'}) , 404

@app.errorhandler(500)

def server\_error(error):

return jsonify({'Error' : 'server error could not response to the request'}) , 500