☰ Documentation Settings ▾

misyncdoc

This is API documentation for given tasks

Task 1

1. **Create API's through which we can add customers and their details. The details to beincluded are:**

● Customer Name
● Email
● Mobile Number
● City
● Customer ID - Should be Auto generated (Unique ID)

Introduction to What API is ? and How I implement Flask here in this Task

An API or Application Programming Interface is a Program used to deliver data or communicate between two different Process of a System . A REST API is an API that conforms to the design principles of the REST, or *representational state transfer* architectural style. For this reason, REST APIs are sometimes referred to RESTful APIs .

An [API] is a mechanism that enables an application or service to access a resource within another application or service .

We can use high level languages to work on APIs like Python with Flask and Django , Java with Spring and log4j , and js with angular and node and react etc .

Flask is a Python based Framework to work on REST API for WebApps .

Flask configs for the current task in Linux .

```
sudo apt-get install pip3 #or pip
```

```
sudo apt-get install python3 #if not there in your machine
```

```
pip3 install flask requests json #or pip
```

```
export FLASK_APP='./main.py'
```

Here the FLASK_APP is the Flask's Global variable which is used for instantiating the Flask Server .

```
export FLASK_ENV=development
```

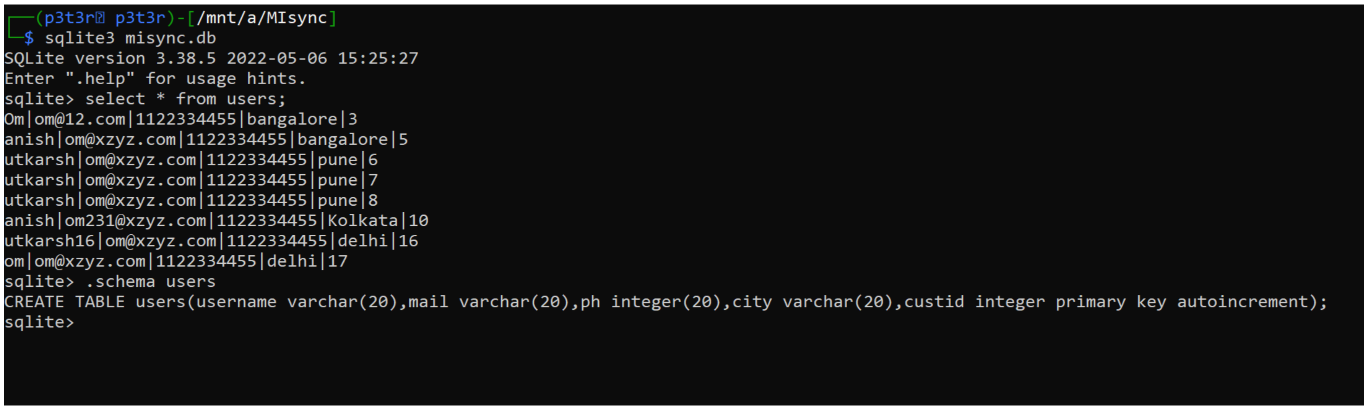
☰ Documentation Settings ▼

`flask run`

will start the Flask Server .

Backend Database

IN THIS DOCUMENTATION SQLITE3 DATABASE IS USED



```
┌──(p3t3r㉿ p3t3r)-[/mnt/a/MIsync]
└─$ sqlite3 misync.db
SQLite version 3.38.5 2022-05-06 15:25:27
Enter ".help" for usage hints.
sqlite> select * from users;
Om|om@12.com|1122334455|bangalore|3
anish|om@xzyz.com|1122334455|bangalore|5
utkarsh|om@xzyz.com|1122334455|pune|6
utkarsh|om@xzyz.com|1122334455|pune|7
utkarsh|om@xzyz.com|1122334455|pune|8
anish|om231@xzyz.com|1122334455|Kolkata|10
utkarsh16|om@xzyz.com|1122334455|delhi|16
om|om@xzyz.com|1122334455|delhi|17
sqlite> .schema users
CREATE TABLE users(username varchar(20),mail varchar(20),ph integer(20),city varchar(20),custid integer primary key autoincrement);
sqlite>
```

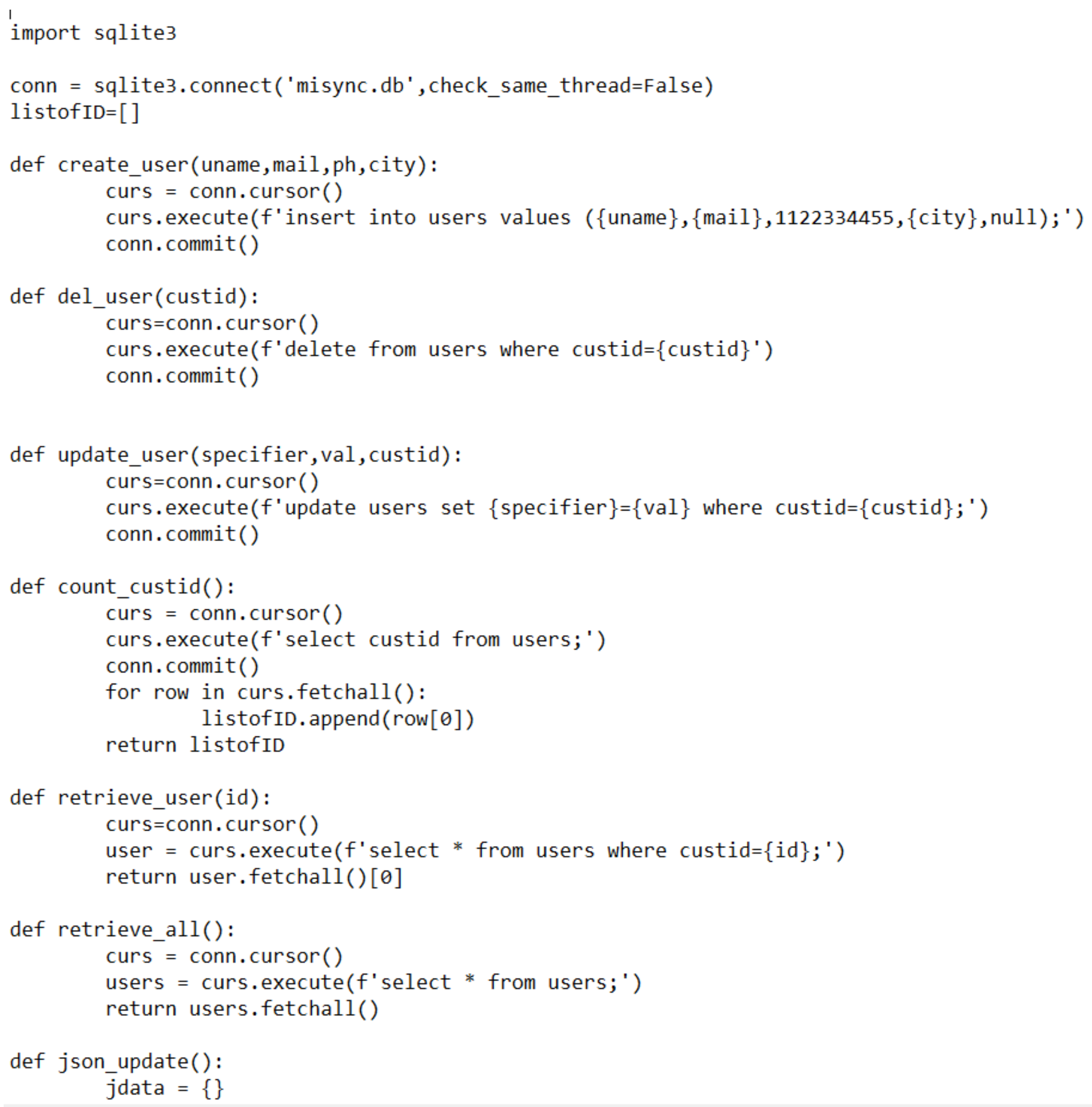
SQLITE3 DATABASE CONNECTION WITH FLASK USING DBCON.PY FILE IN MAIN.PY ,

`import sqlite3`

`from dbcon import *`

dbcon.py

Documentation Settings

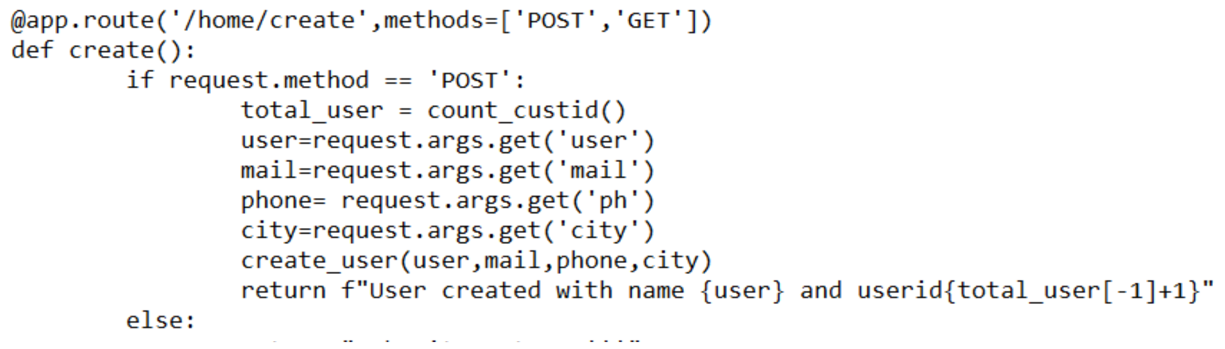


```
import sqlite3

conn = sqlite3.connect('misync.db',check_same_thread=False)
listofID=[]

def create_user(uname,mail,ph,city):
        curs = conn.cursor()
        curs.execute(f'insert into users values ({uname},{mail},1122334455,{city},null);')
        conn.commit()

def del_user(custid):
        curs=conn.cursor()
        curs.execute(f'delete from users where custid={custid}')
        conn.commit()


def update_user(specifier,val,custid):
        curs=conn.cursor()
        curs.execute(f'update users set {specifier}={val} where custid={custid};')
        conn.commit()

def count_custid():
        curs = conn.cursor()
        curs.execute(f'select custid from users;')
        conn.commit()
        for row in curs.fetchall():
                listofID.append(row[0])
        return listofID

def retrieve_user(id):
        curs=conn.cursor()
        user = curs.execute(f'select * from users where custid={id};')
        return user.fetchall()[0]

def retrieve_all():
        curs = conn.cursor()
        users = curs.execute(f'select * from users;')
        return users.fetchall()

def json_update():
        jdata = {}
```

Backend Flask Route Methods using Python

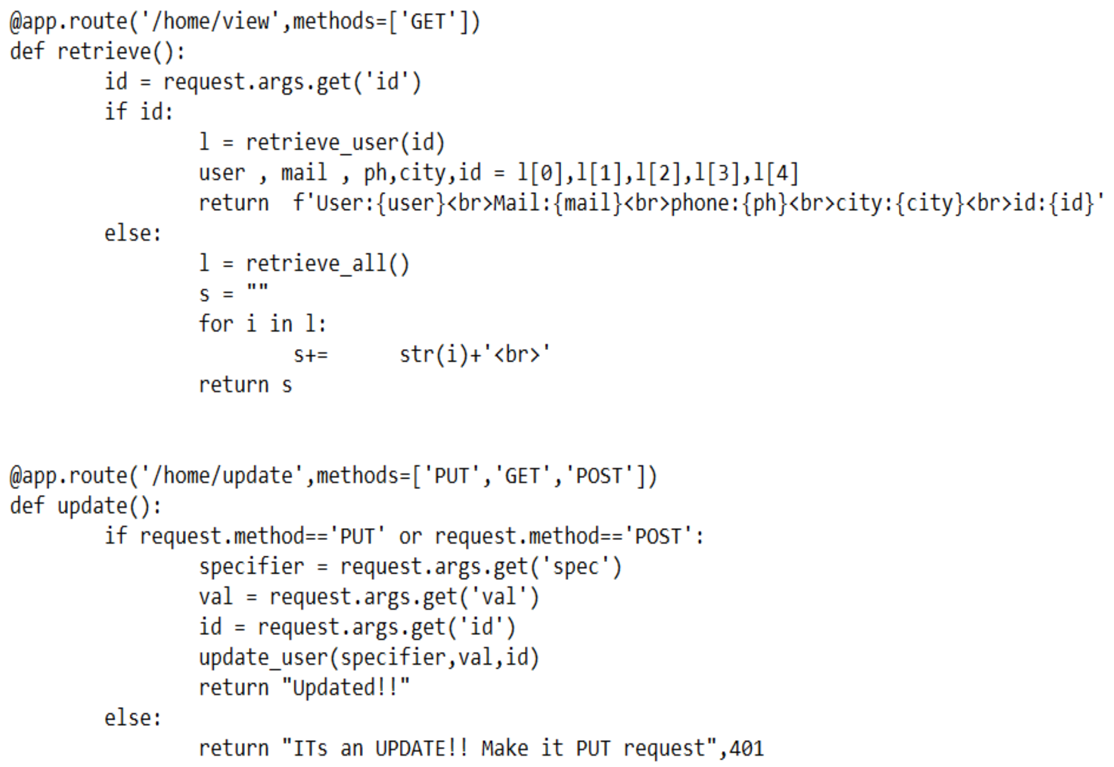
I HAVE USED 4 METHODS EACH FOR EACH OPERATION AND ITS RESPECTIVE ROUTE

Create Method



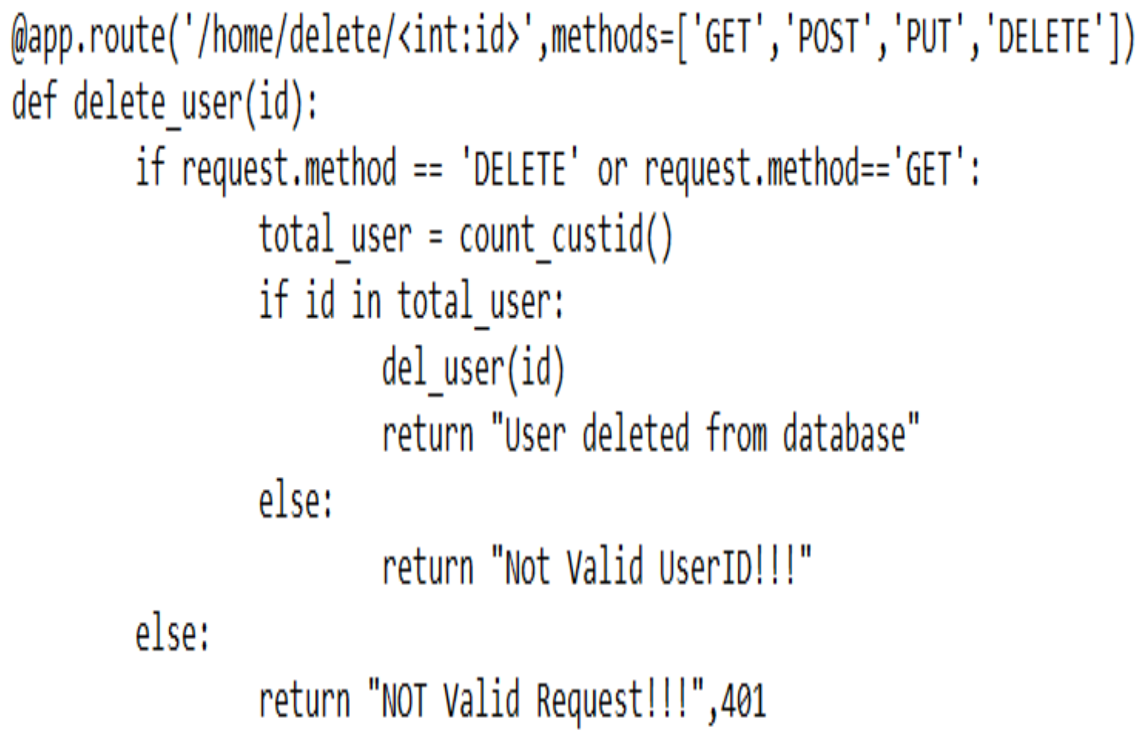
```
@app.route('/home/create',methods=['POST','GET'])
def create():
        if request.method == 'POST':
                total_user = count_custid()
                user=request.args.get('user')
                mail=request.args.get('mail')
                phone= request.args.get('ph')
                city=request.args.get('city')
                create_user(user,mail,phone,city)
                return f"User created with name {user} and userid{total_user[-1]+1}"
        else:
```

Documentation Settings



```
@app.route('/home/view',methods=['GET'])
def retrieve():
        id = request.args.get('id')
        if id:
                l = retrieve_user(id)
                user , mail , ph,city,id = l[0],l[1],l[2],l[3],l[4]
                return  f'User:{user}<br>Mail:{mail}<br>phone:{ph}<br>city:{city}<br>id:{id}'
        else:
                l = retrieve_all()
                s = ""
                for i in l:
                        s+=      str(i)+'<br>'
                return s


@app.route('/home/update',methods=['PUT','GET','POST'])
def update():
        if request.method=='PUT' or request.method=='POST':
                specifier = request.args.get('spec')
                val = request.args.get('val')
                id = request.args.get('id')
                update_user(specifier,val,id)
                return "Updated!!"
        else:
                return "ITs an UPDATE!! Make it PUT request",401
```

Delete



```
@app.route('/home/delete/<int:id>',methods=['GET','POST','PUT','DELETE'])
def delete_user(id):
        if request.method == 'DELETE' or request.method=='GET':
                total_user = count_custid()
                if id in total_user:
                        del_user(id)
                        return "User deleted from database"
                else:
                        return "Not Valid UserID!!!"
        else:
                return "NOT Valid Request!!!",401
```

☰ Documentation Settings ▾

```
127.0.0.1:5000/home/create?user='om'&mail='om@xzyz.com'&ph=0011223344&city='delhi'
```

CRUD (Create)

Create Request

We make POST request to perform Create Operation using API

PARAMS

user

'om'

mail

'om@xzyz.com'

ph

0011223344

city

'delhi'

Example Request Create

```
curl --location --request POST '127.0.0.1:5000/home/create?user='\''om'\''&mail='\''om@xzyz.com'\''&ph=
```

Example Response 200 OK

Body Header (5)

```
User created with name 'om' and userid17
```

GET Retrieval

Documentation Settings

CRUD operation (Retrieval/View)

View / Retrieval

This Request shows the View of the Database or Retrieval of Database

We haven't jsonify the Output of our requests , so its showing in DOM formatted in the Response .

Example Request Retrieval

```
curl --location --request GET '127.0.0.1:5000/home/view'
```

Example Response 200 OK

Body Header (5)

```
('Om', 'om@12.com', 1122334455, 'bangalore', 3)
<br>('anish', 'om@xzyz.com', 1122334455, 'bangalore', 5)
<br>('utkarsh', 'om@xzyz.com', 1122334455, 'pune', 6)
<br>('utkarsh', 'om@xzyz.com', 1122334455, 'pune', 7)
<br>('utkarsh', 'om@xzyz.com', 1122334455, 'pune', 8)
<br>('Om', 'om231@xzyz.com', 1122334455, 'Kolkata', 10)
<br>('utkarsh16', 'om@xzyz.com', 1122334455, 'delhi', 16)
<br>('om', 'om@xzyz.com', 1122334455, 'delhi', 17)
<br>
```

PUT Update

127.0.0.1:5000/home/update?spec='username'&val='anish'&id=10

PARAMS

spec

'username'

val

Documentation Settings

id

10

Example Request

Update

```
curl --location --request PUT '127.0.0.1:5000/home/update?spec='\''username'\''&val='\''anish'\''&id=1(
```

Example Response

200 OK

Body Header (5)

```
Updated!!
```

GET Delete

http://127.0.0.1:5000/home/delete/15

Documentation Settings