from \_\_future\_\_ import division

from flask\_restful import Resource

from flask import jsonify, request

from models import DealerMobileUpdate,ProdinvWH,Sup\_Upload1,PriceLists,atr,ConfHeader,ConfDetails,ProductOffer,ProductWhatNew,Reviews\_info,Category,Sub\_Category,CustomerDetails,CustBillingAddress,CustShippingAddress,WishList,\

OrderitemsSite,InitiateOrdersSite,OrderitemsSite,ProdInvShop,CustomerOrders,OrderCoupons,InitiateOrders,Orderitems,CompareProducts,HedaerData,Scheme,LatestRfqNum,DealerOrderData,Product\_Info,Reg\_Dealer

import json,ast

import datetime

from time import strftime

import datetime

import os

from pathlib import Path

from mongoengine.queryset.visitor import Q

import collections

from collections import Counter

from datetime import datetime,timedelta

import requests

from multiprocessing import Pool

import requests,urllib

import itertools as it

import random

import threading

import functools

from flask\_httpauth import HTTPBasicAuth

from datetime import date

import onetimepass as otp

secret\_key = '4r5t@W'

auth = HTTPBasicAuth()

USER\_DATA = {

"admin": "SuperSecretPwd"

}

def sendMail(email,subject,htmlbody):

print email

value = requests.post("https://api.mailgun.net/v3/mg.shopmytools.com/messages",

auth=("api", "key-8bee70be27164355bdf6ad96ba54ca6d"),

data={"from": 'SMT-Hub <info@mg.shopmytools.com>',

"to": email,

"subject": subject,

"html": htmlbody})

def SMS(message,mobile):

print mobile

url = "http://www.smscountry.com/smscwebservice\_bulk.aspx"

values = {'user' : 'powertex',

'passwd' : 'Powertexgst1',

'message':message,

'mobilenumber':mobile,

'mtype':'N',

'DR':'Y'

}

data=urllib.urlencode(values)

data = data.encode('utf-8')

f = urllib.urlopen(url, data)

class HomePageData(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args = request.args['link']

from\_=request.args['from']

if args=='' and from\_=='':

header\_data=HedaerData.objects(site\_name='SMT')

sitenbr = json.loads(header\_data.to\_json())

header=ConfDetails.objects(types='Headerlogo',link='Smt').order\_by('-id')

print header

headerlogo = json.loads(header.to\_json())

banner=ConfDetails.objects(types='Banner',link='Smt').order\_by('-id')

bannerdata = json.loads(banner.to\_json())

details1=ConfDetails.objects(types='Details1',link='Smt').order\_by('-id')

details1data = json.loads(details1.to\_json())

details2=ConfDetails.objects(types='Details2',link='Smt').order\_by('-id')

details2data = json.loads(details2.to\_json())

details3=ConfDetails.objects(types='Details3',link='Smt').order\_by('-id')

details3data = json.loads(details3.to\_json())

footer1=ConfDetails.objects(types='Footer',link='Smt',sl\_num\_\_in=['1','2','3']).order\_by('-id')

footer1data = json.loads(footer1.to\_json())

footer2=ConfDetails.objects(types='Footer',link='Smt',sl\_num\_\_in=['4','5','6']).order\_by('-id')

footer2data = json.loads(footer2.to\_json())

return {'status':'Success','SitePhnNbr':sitenbr,'HeaderLogo':headerlogo,'BannerData':bannerdata,'Details1':details1data,'Details2':details2data,'Details3':details3data,'Footer':footer1data,'Footer2':footer2data }

date=datetime.strptime(from\_, '%d-%m-%Y')

cuurent\_date=date.date()

header\_data=HedaerData.objects(site\_name='Powertex')

sitenbr = json.loads(header\_data.to\_json())

header=ConfDetails.objects(types='Headerlogo',link=args)#,from\_date\_\_lte=from\_,to\_date\_\_gte=from\_).order\_by('-id')

headerlogo = json.loads(header.to\_json())

header\_list=[]

banner\_list=[]

details1\_list=[]

details2\_list=[]

details3\_list=[]

footer1\_list=[]

footer2\_list=[]

for name in headerlogo:

fromdate=datetime.strptime(name['from\_date'], '%d-%m-%Y')

frm\_date= fromdate.date()

todate=datetime.strptime(name['to\_date'], '%d-%m-%Y')

t\_date= todate.date()

if frm\_date<=cuurent\_date and t\_date>=cuurent\_date:

json\_output= json.loads(json.dumps(name))

header\_list.append(json\_output)

banner=ConfDetails.objects(types='Banner',link=args)

headerlogo = json.loads(banner.to\_json())

for name in headerlogo:

fromdate=datetime.strptime(name['from\_date'], '%d-%m-%Y')

frm\_date= fromdate.date()

todate=datetime.strptime(name['to\_date'], '%d-%m-%Y')

t\_date= todate.date()

if frm\_date<=cuurent\_date and t\_date>=cuurent\_date:

json\_output= json.loads(json.dumps(name))

banner\_list.append(json\_output)

details1=ConfDetails.objects(types='Details1',link=args)

headerlogo = json.loads(details1.to\_json())

for name in headerlogo:

fromdate=datetime.strptime(name['from\_date'], '%d-%m-%Y')

frm\_date= fromdate.date()

todate=datetime.strptime(name['to\_date'], '%d-%m-%Y')

t\_date= todate.date()

if frm\_date<=cuurent\_date and t\_date>=cuurent\_date:

json\_output= json.loads(json.dumps(name))

details1\_list.append(json\_output)

details2=ConfDetails.objects(types='Details2',link=args)

headerlogo = json.loads(details2.to\_json())

for name in headerlogo:

fromdate=datetime.strptime(name['from\_date'], '%d-%m-%Y')

frm\_date= fromdate.date()

todate=datetime.strptime(name['to\_date'], '%d-%m-%Y')

t\_date= todate.date()

if frm\_date<=cuurent\_date and t\_date>=cuurent\_date:

json\_output= json.loads(json.dumps(name))

details2\_list.append(json\_output)

details3=ConfDetails.objects(types='Details3',link=args)

headerlogo = json.loads(details3.to\_json())

for name in headerlogo:

fromdate=datetime.strptime(name['from\_date'], '%d-%m-%Y')

frm\_date= fromdate.date()

todate=datetime.strptime(name['to\_date'], '%d-%m-%Y')

t\_date= todate.date()

if frm\_date<=cuurent\_date and t\_date>=cuurent\_date:

json\_output= json.loads(json.dumps(name))

details3\_list.append(json\_output)

footer1=ConfDetails.objects(types='Details3',link=args,sl\_num\_\_in=['1','2','3'])

headerlogo = json.loads(footer1.to\_json())

for name in headerlogo:

fromdate=datetime.strptime(name['from\_date'], '%d-%m-%Y')

frm\_date= fromdate.date()

todate=datetime.strptime(name['to\_date'], '%d-%m-%Y')

t\_date= todate.date()

if frm\_date<=cuurent\_date and t\_date>=cuurent\_date:

json\_output= json.loads(json.dumps(name))

footer1\_list.append(json\_output)

footer2=ConfDetails.objects(types='Details3',link=args,sl\_num\_\_in=['4','5','6'])

headerlogo = json.loads(footer2.to\_json())

for name in headerlogo:

fromdate=datetime.strptime(name['from\_date'], '%d-%m-%Y')

frm\_date= fromdate.date()

todate=datetime.strptime(name['to\_date'], '%d-%m-%Y')

t\_date= todate.date()

if frm\_date<=cuurent\_date and t\_date>=cuurent\_date:

json\_output= json.loads(json.dumps(name))

footer2\_list.append(json\_output)

return {'status':'Success','SitePhnNbr':sitenbr,'HeaderLogo':header\_list,'BannerData':banner\_list,'Details1':details1\_list,'Details2':details2\_list,'Details3':details3\_list,'Footer':footer1\_list,'Footer2':footer2\_list }

class OffersProducts(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args = request.args['upload\_subcategory']

offerdata=ProductOffer.objects(sub\_category\_\_icontains=args)

files = os.listdir("./static/ProductImages")

prod\_data=[]

for prod in offerdata:

prod\_details={}

prod\_details['productname']=prod.productname

prod\_details['sl\_num']=prod.sl\_num

prod\_details['from\_date']=prod.from\_date

prod\_details['to\_date']=prod.to\_date

prod\_details['mrp']=prod.mrp

prod\_details['discount']=prod.discount

prod\_details['disc\_price']=prod.disc\_price

prod\_details['category']=prod.category

prod\_details['sub\_category']=prod.sub\_category

prod\_details['productid']=prod.productid

total\_prod='011'+prod.productid+'.jpg'

for prods in files:

if prods==total\_prod:

prod\_details['image']=prods

print prod\_details['image']

prod\_data.append(prod\_details)

return {'status':'Success',"offer\_products":prod\_data}

class NewArrivalsProducts(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args = request.args['upload\_subcategory']

newarrival=ProductWhatNew.objects(sub\_category\_\_icontains=args)

files = os.listdir("./static/ProductImages")

prod\_data=[]

for prod in newarrival:

prod\_details={}

prod\_details['category']=prod.category

prod\_details['sub\_category']=prod.sub\_category

prod\_details['productname']=prod.productname

prod\_details['productid']=prod.productid

name\_compare=Sup\_Upload1.objects.get(upload\_name=prod\_details['productname'])

prod\_details['upload\_mrp']=name\_compare.upload\_mrp

prod\_details['percentage']=name\_compare.end\_user\_disc

disc=float(prod\_details['upload\_mrp'])-(float(prod\_details['upload\_mrp'])\*float(prod\_details['percentage'])/100)

prod\_details['enduser\_price']=disc

total\_prod='011'+prod.productid+'.jpg'

for prods in files:

if prods==total\_prod:

prod\_details['image']=prods

prod\_data.append(prod\_details)

return {'status':'Success',"Newarrivals\_products":prod\_data}

class ProductDetails(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args=request.args

product=args['product\_name']

realted\_prod\_list=[]

upsell\_prod\_list=[]

sup = Sup\_Upload1.objects.get(upload\_name=product)

files = os.listdir("./static/ProductImages")

if sup:

details={}

details['upload\_name']=sup.upload\_name

details['upload\_brand']=sup.upload\_brand

details['description']=sup.description

details['upload\_modelno']=sup.upload\_modelno

details['upload\_netWeight']=sup.upload\_netWeight

details['avgrating']=sup.avgrating

details['upload\_category']=sup.upload\_category

details['upload\_subcategory']=sup.upload\_subcategory

details['key\_features']=sup.key\_features

details['prod\_functions']=sup.prod\_functions

details['upload\_mrp']=sup.upload\_mrp

details['percentage']=sup.end\_user\_disc

disc=float(details['upload\_mrp'])-(float(details['upload\_mrp'])\*float(details['percentage'])/100)

details['enduser\_price']=disc

total\_prod='021'+sup.upload\_id+'.jpg'

for prods in files:

if prods==total\_prod:

details['image']=prods

datas=[]

for j in sup.attributes:

details1={}

details1['atrname']=j.atrname

details1['atrvalue']=j.atrvalue

test=json.loads(json.dumps(details1))

datas.append(test)

details2={}

images={}

images\_list=['011'+sup.upload\_id+'.jpg','021'+sup.upload\_id+'.jpg','031'+sup.upload\_id+'.jpg','041'+sup.upload\_id+'.jpg','051'+sup.upload\_id+'.jpg','061'+sup.upload\_id+'.jpg',]

lists=[]

for images in files:

for image in images\_list:

if image==images:

image=images

lists.append(image)

imageshigh={}

images\_list\_high=['012'+sup.upload\_id+'.jpg','022'+sup.upload\_id+'.jpg','032'+sup.upload\_id+'.jpg','042'+sup.upload\_id+'.jpg','052'+sup.upload\_id+'.jpg','062'+sup.upload\_id+'.jpg']

lists\_high=[]

for imageshigh in files:

for image in images\_list\_high:

if image==imageshigh:

image=imageshigh

lists\_high.append(image)

for productsQuerySets in Sup\_Upload1.objects(upload\_category=sup.upload\_category,upload\_subcategory=sup.upload\_subcategory,upload\_brand=sup.upload\_brand,upload\_name\_\_ne=product):

files = os.listdir("./static/ProductImages")

if productsQuerySets:

related\_prod\_info={}

related\_prod\_info['upload\_name']=productsQuerySets.upload\_name

related\_prod\_info['upload\_id']=productsQuerySets.upload\_id

related\_prod\_info['avgrating']=productsQuerySets.avgrating

related\_prod\_info['upload\_category']=productsQuerySets.upload\_category

related\_prod\_info['upload\_subcategory']=productsQuerySets.upload\_subcategory

related\_prod\_info['upload\_mrp']=productsQuerySets.upload\_mrp

related\_prod\_info['percentage']=productsQuerySets.end\_user\_disc

disc=float(details['upload\_mrp'])-(float(related\_prod\_info['upload\_mrp'])\*float(related\_prod\_info['percentage'])/100)

related\_prod\_info['enduser\_price']=disc

image='011'+productsQuerySets.upload\_id+'.jpg'

for prods in files:

if prods==image:

related\_prod\_info['image']=prods

realted\_prod\_list.append(related\_prod\_info)

pricesMinMax = Sup\_Upload1.objects(Q(upload\_category=sup.upload\_category)| Q(upload\_subcategory=sup.upload\_subcategory)| Q(upload\_brand=sup.upload\_brand),upload\_name\_\_ne=product,status='Accept')

minimum = json.loads(pricesMinMax.to\_json())

rev=[]

final\_list=[]

for upsell\_productsQuerySets in Sup\_Upload1.objects(upload\_category=sup.upload\_category,upload\_subcategory=sup.upload\_subcategory,upload\_brand=sup.upload\_brand,upload\_name\_\_ne=product):

#print upsell\_productsQuerySets.prices[0]['doubleoffer\_price']

files = os.listdir("./static/ProductImages")

if upsell\_productsQuerySets:

upsell\_prod\_info={}

upsell\_prod\_info['upload\_name']=upsell\_productsQuerySets.upload\_name

upsell\_prod\_info['upload\_id']=upsell\_productsQuerySets.upload\_id

upsell\_prod\_info['avgrating']=upsell\_productsQuerySets.avgrating

upsell\_prod\_info['upload\_category']=upsell\_productsQuerySets.upload\_category

upsell\_prod\_info['upload\_subcategory']=upsell\_productsQuerySets.upload\_subcategory

upsell\_prod\_info['upload\_mrp']=upsell\_productsQuerySets.upload\_mrp

upsell\_prod\_info['percentage']=upsell\_productsQuerySets.end\_user\_disc

disc=float(upsell\_prod\_info['upload\_mrp'])-(float(upsell\_prod\_info['upload\_mrp'])\*float(upsell\_prod\_info['percentage'])/100)

upsell\_prod\_info['enduser\_price']=disc

image='011'+upsell\_productsQuerySets.upload\_id+'.jpg'

for prods in files:

if prods==image:

upsell\_prod\_info['image']=prods

upsell\_prod\_list.append(upsell\_prod\_info)

prdr=Reviews\_info.objects(prod\_desc=args['product\_name'],status='Active')

for reviews in prdr():

data={}

data['user']=reviews.user

data['created\_date']=str(reviews.created\_date)

data['review']=reviews.review

data['rating']=reviews.rating

data['rating\_comments']=reviews.rating\_comments

details\_json=json.loads(json.dumps(data))

rev.append(details\_json)

prod\_rev=Reviews\_info.objects(prod\_desc=product,status='Active').order\_by('-rating')

rev\_count=[]

for reviews in prod\_rev():

rev\_count.append(reviews.rating)

rev\_data=collections.Counter(rev\_count)

for (k,v) in rev\_data.items():

review={}

review['count']=v

reviews\_data=Reviews\_info.objects(prod\_desc=product,rating=k,status='Active').order\_by('-rating')

info=[]

for reviews in reviews\_data:

data={}

data['user']=reviews.user

data['created\_date']=str(reviews.created\_date)

data['email']=reviews.email

data['rating']=reviews.rating

data['prod\_desc']=reviews.prod\_desc

data['rating\_comments']=reviews.rating\_comments

detail\_json=json.loads(json.dumps(data))

info.append(detail\_json)

review['data']=info

final\_list.append(review)

return {"status":"success","Product":details,"attribute\_info":datas,"price\_info":details2,"Low\_Images":lists,"High\_Images":lists\_high,"Related\_Products":realted\_prod\_list,"Upsell\_products":upsell\_prod\_list,"product\_Reviews":rev,"Review\_info":final\_list }#"Upsell products":upsell\_prod\_list}

class Reviews(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data = request.get\_json(force=True)

userid=json\_data['user\_id']

print userid

prod\_desc=json\_data['prod\_desc']

rating=json\_data['rating']

mobile\_number='91'+json\_data['mobile\_number']

print json\_data['mobile\_number']

review=json\_data['review']

rating\_comments=json\_data['rating\_comments']

user =json\_data['user\_name']

if '@' in userid:

cust\_checking=CustomerDetails.objects(email=json\_data['user\_id'])

if cust\_checking.count()==0:

return {"status":"user not registered"}

else:

user\_info=Reviews\_info.objects(email=userid,prod\_desc=prod\_desc)

if user\_info.count()==0:

#return {"status":"user not registered"}

info=Reviews\_info(email=userid,prod\_desc=prod\_desc,rating=rating,mobile\_number=mobile\_number,review=review,rating\_comments=rating\_comments,user=user,mobile='')

info.save()

else:

review\_info=Reviews\_info.objects.get(email=userid,prod\_desc=prod\_desc)

review\_info.rating\_comments=rating\_comments

review\_info.prod\_desc=prod\_desc

review\_info.rating=rating

review\_info.review=review

review\_info.save()

else:

cust\_checking=CustomerDetails.objects(mobile='91'+json\_data['user\_id'])

if cust\_checking.count()==0:

return {"status":"user not registered"}

else:

user\_info=Reviews\_info.objects(mobile=userid,prod\_desc=prod\_desc)

if user\_info.count()==0:

info=Reviews\_info(email='',prod\_desc=prod\_desc,rating=rating,mobile\_number=mobile\_number,review=review,rating\_comments=rating\_comments,user=user,mobile=userid)

info.save()

else:

review\_data=Reviews\_info.objects.get(mobile=userid,prod\_desc=prod\_desc)

review\_data.rating\_comments=rating\_comments

review\_data.prod\_desc=prod\_desc

review\_data.rating=rating

review\_data.review=review

review\_data.save()

return {"success":"success"}

class ProductlistService(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

#args = request.args

json\_data = request.get\_json(force=True)

files = os.listdir("./static/ProductImages")

#search items ma

category = json\_data['category'] if 'category' in json\_data else ''

print category

cat= type(category)

#print cat

fromval=json\_data['from'] if 'from' in json\_data else '0'

to=json\_data['to'] if 'to' in json\_data else '11'

view=json\_data['view'] if 'view' in json\_data else ''

#search optional

subcategory= json\_data['subcategory'] if 'subcategory' in json\_data else ['']

print subcategory

brand= json\_data['brand'] if 'brand' in json\_data else ['']

print

pricerange= json\_data['pricerange'] if 'pricerange' in json\_data else ''

#remove duplicate

sub\_list=list(set(subcategory))

brands\_list=list(set(brand))

#filter/orderby data

val= json\_data['val'] if 'val' in json\_data else False

filterStr='-id'

if val=="topselling":

filterStr='-salesqty'

if val=="popularty":

filterStr='-avgrating'

if val=="namefilter":

filterStr='+upload\_name'

if val=="price\_low\_high":

filterStr='+upload\_mrp'

if val=="price\_high\_low":

filterStr='-upload\_mrp'

if brand!=[''] and category=='':

productsQuerySet=Sup\_Upload1.objects(upload\_brand\_\_in=brand).only('salesqty','avgrating','upload\_id','upload\_name','upload\_category','upload\_subcategory','upload\_mrp','end\_user\_disc').order\_by(filterStr)[int(fromval):int(to)]

print productsQuerySet,'first'

if productsQuerySet.count()==0:

return {"status":"No data avialbale"}

lists=[]

products = json.loads(productsQuerySet.to\_json())

for prod in products:

print prod

print prod['upload\_id'],'upload\_id'

prod\_inf={}

total\_prod='011'+prod['upload\_id']+'.jpg'

total\_info=prod

for prods in files:

if prods==total\_prod:

print prods,'prods'

prod\_inf['product']=prod

prod\_inf['image']=prods

disc=float(prod['upload\_mrp'])-(float(prod['upload\_mrp'])\*float(prod['end\_user\_disc'])/100)

prod\_inf['enduser\_price']=disc

lists.append(prod\_inf)

value=[x for x in lists if x]

catsub=[]

for subcat in Sup\_Upload1.objects(upload\_brand\_\_in=brand):

data={}

data['category']=subcat.upload\_category

data['subcategory']=subcat.upload\_subcategory

inlistFlag = True

for cat in catsub:

if(cat['subcategory']==subcat.upload\_subcategory):

if(cat['category']==subcat.upload\_category):

inlistFlag=False

if inlistFlag:

catsub.append(data)

subCategories=sorted(catsub)

subCategory = Sup\_Upload1.objects(upload\_brand\_\_in=brand).only('upload\_subcategory').distinct('upload\_subcategory')

subcategories = json.loads(json.dumps(subCategory))

mobilesubCategories=sorted(subcategories)

return {"status":"success","value":value,"mobilesubcategories":mobilesubCategories,"from":fromval,"to":to,'filterdata':{'category':category,'subcategory':sub\_list,'brand':brands\_list,'pricerange':pricerange}}

elif subcategory==[''] and brand==[''] and pricerange=='':

#print datetime.datetime.now().strftime("%H:%M:%S.%f")[0:-3]

productsQuerySet=Sup\_Upload1.objects(upload\_category=category).only('upload\_id','salesqty','avgrating','upload\_name','upload\_brand','upload\_subcategory','upload\_photo','upload\_mrp','end\_user\_disc').order\_by(filterStr)[int(fromval):int(to)]

print productsQuerySet,'second'

if productsQuerySet.count()==0:

return {"status":"No data avialbale"}

lists=[]

products = json.loads(productsQuerySet.to\_json())

for prod in products:

print prod

print prod['upload\_id'],'upload\_id'

prod\_inf={}

total\_prod='011'+prod['upload\_id']+'.jpg'

total\_info=prod

for prods in files:

if prods==total\_prod:

print prods,'prods'

prod\_inf['product']=prod

prod\_inf['image']=prods

disc=float(prod['upload\_mrp'])-(float(prod['upload\_mrp'])\*float(prod['end\_user\_disc'])/100)

prod\_inf['enduser\_price']=disc

lists.append(prod\_inf)

value=[x for x in lists if x]

subCategory = Sup\_Upload1.objects(upload\_category=category).only('upload\_subcategory').distinct('upload\_subcategory')

subcategories = json.loads(json.dumps(subCategory))

subCategories=sorted(subcategories)

brands = Sup\_Upload1.objects(upload\_category=category).only('upload\_brand').distinct('upload\_brand')# it returns the list of brands, remove duplicates

branddata = json.loads(json.dumps(brands))

brandslist=sorted(branddata)

return {"status":"success","value":value,"from":fromval,"to":to,'view':view,'filterdata':{'category':category,'subcategory':sub\_list,'brand':brands\_list,'pricerange':pricerange,'sortdata':val}}

elif subcategory!=[''] and brand==[''] and pricerange=='':

#print datetime.datetime.now().strftime("%H:%M:%S.%f")[0:-3]

productsQuerySet=Sup\_Upload1.objects(upload\_category=category,upload\_subcategory\_\_in=subcategory).only('salesqty','avgrating','upload\_id','upload\_name','upload\_category','upload\_subcategory','upload\_mrp','end\_user\_disc').order\_by(filterStr)[int(fromval):int(to)]

print productsQuerySet,'thrid'

if productsQuerySet.count()==0:

return {"status":"No data avialbale"}

lists=[]

products = json.loads(productsQuerySet.to\_json())

for prod in products:

print prod

print prod['upload\_id'],'upload\_id'

prod\_inf={}

total\_prod='011'+prod['upload\_id']+'.jpg'

total\_info=prod

for prods in files:

if prods==total\_prod:

print prods,'prods'

prod\_inf['product']=prod

prod\_inf['image']=prods

disc=float(prod['upload\_mrp'])-(float(prod['upload\_mrp'])\*float(prod['end\_user\_disc'])/100)

prod\_inf['enduser\_price']=disc

lists.append(prod\_inf)

value=[x for x in lists if x]

subCategory = Sup\_Upload1.objects(upload\_category=category).only('upload\_subcategory').distinct('upload\_subcategory')

subcategories = json.loads(json.dumps(subCategory))

mobilesubCategories=sorted(subcategories)

brands = Sup\_Upload1.objects(upload\_category=category).only('upload\_brand').distinct('upload\_brand')# it returns the list of brands, remove duplicates

branddata = json.loads(json.dumps(brands))

mobilebrandslist=sorted(branddata)

totalCagetoryProducts = Sup\_Upload1.objects(upload\_category=category,upload\_subcategory\_\_in=subcategory).count()

return {"status":"success","value":value,"from":fromval,"to":to,'filterdata':{'category':category,'subcategory':sub\_list,'brand':brands\_list,'pricerange':pricerange}}

elif subcategory!=[''] and brand!=[''] and pricerange=='':

#print datetime.datetime.now().strftime("%H:%M:%S.%f")[0:-3]

productsQuerySet=Sup\_Upload1.objects(upload\_category=category,upload\_subcategory\_\_in=subcategory,upload\_brand\_\_in = brand).only('salesqty','avgrating','upload\_id','upload\_name','upload\_category','upload\_subcategory','upload\_mrp','end\_user\_disc').order\_by(filterStr)[int(fromval):int(to)]

print productsQuerySet,'fourth'

if productsQuerySet.count()==0:

return {"status":"No data avialbale"}

lists=[]

products = json.loads(productsQuerySet.to\_json())

for prod in products:

print prod

print prod['upload\_id'],'upload\_id'

prod\_inf={}

total\_prod='011'+prod['upload\_id']+'.jpg'

total\_info=prod

for prods in files:

if prods==total\_prod:

print prods,'prods'

prod\_inf['product']=prod

prod\_inf['image']=prods

disc=float(prod['upload\_mrp'])-(float(prod['upload\_mrp'])\*float(prod['end\_user\_disc'])/100)

prod\_inf['enduser\_price']=disc

lists.append(prod\_inf)

value=[x for x in lists if x]

totalCagetoryProducts = Sup\_Upload1.objects(upload\_category=category,upload\_subcategory\_\_in=subcategory,upload\_brand\_\_in = brand).count()

#----------mobile

subCategory = Sup\_Upload1.objects(upload\_category=category).only('upload\_subcategory').distinct('upload\_subcategory')

subcategories = json.loads(json.dumps(subCategory))

mobilesubCategories=sorted(subcategories)

brands = Sup\_Upload1.objects(upload\_category=category).only('upload\_brand').distinct('upload\_brand')# it returns the list of brands, remove duplicates

branddata = json.loads(json.dumps(brands))

mobilebrandslist=sorted(branddata)

return {"status":"success","value":value,"mobilesubcategories":mobilesubCategories,"from":fromval,"to":to,'filterdata':{'category':category,'subcategory':sub\_list,'brand':brands\_list,'pricerange':pricerange}}

elif subcategory==[''] and brand!=[''] and pricerange=='':

#print datetime.datetime.now().strftime("%H:%M:%S.%f")[0:-3]

productsQuerySet=Sup\_Upload1.objects(upload\_category=category,upload\_brand\_\_in = brand).only('salesqty','avgrating','upload\_id','upload\_name','upload\_category','upload\_subcategory','upload\_mrp','end\_user\_disc').order\_by(filterStr)[int(fromval):int(to)]

print productsQuerySet,'fivth'

if productsQuerySet.count()==0:

return {"status":"No data avialbale"}

lists=[]

products = json.loads(productsQuerySet.to\_json())

for prod in products:

print prod

print prod['upload\_id'],'upload\_id'

prod\_inf={}

total\_prod='011'+prod['upload\_id']+'.jpg'

total\_info=prod

for prods in files:

if prods==total\_prod:

print prods,'prods'

prod\_inf['product']=prod

prod\_inf['image']=prods

disc=float(prod['upload\_mrp'])-(float(prod['upload\_mrp'])\*float(prod['end\_user\_disc'])/100)

prod\_inf['enduser\_price']=disc

lists.append(prod\_inf)

value=[x for x in lists if x]

totalCagetoryProducts = Sup\_Upload1.objects(upload\_category=category,upload\_brand = brand).count()

#----------mobile

subCategory = Sup\_Upload1.objects(upload\_category=category).only('upload\_subcategory').distinct('upload\_subcategory')

subcategories = json.loads(json.dumps(subCategory))

mobilesubCategories=sorted(subcategories)

brands = Sup\_Upload1.objects(upload\_category=category).only('upload\_brand').distinct('upload\_brand')# it returns the list of brands, remove duplicates

branddata = json.loads(json.dumps(brands))

mobilebrandslist=sorted(branddata)

return {"status":"success","value":value,"mobilesubcategories":mobilesubCategories,"from":fromval,"to":to,'filterdata':{'category':category,'subcategory':sub\_list,'brand':brands\_list,'pricerange':pricerange}}

elif subcategory!=[''] and brand!=[''] and pricerange!='':

if '-' in pricerange:

pricerange\_lw=pricerange.split('-')[0]

pricerange\_hg=pricerange.split('-')[1]

print type(pricerange\_lw)

#print datetime.datetime.now().strftime("%H:%M:%S.%f")[0:-3]

productsQuerySet=Sup\_Upload1.objects(upload\_category=category,upload\_subcategory\_\_in=subcategory,upload\_brand\_\_in = brand,upload\_mrp\_\_gte=float(pricerange\_lw),upload\_mrp\_\_lte=float(pricerange\_hg)).only('salesqty','avgrating','upload\_name','upload\_id','upload\_brand','upload\_subcategory','upload\_mrp','end\_user\_disc').order\_by(filterStr)[int(fromval):int(to)]

print productsQuerySet,'sixth'

if productsQuerySet.count()==0:

return {"status":"No data avialbale"}

lists=[]

products = json.loads(productsQuerySet.to\_json())

for prod in products:

print prod['upload\_id'],'upload\_id'

prod\_inf={}

total\_prod='011'+prod['upload\_id']+'.jpg'

total\_info=prod

for prods in files:

if prods==total\_prod:

print prods,'prods'

prod\_inf['product']=prod

prod\_inf['image']=prods

disc=float(prod['upload\_mrp'])-(float(prod['upload\_mrp'])\*float(prod['end\_user\_disc'])/100)

prod\_inf['enduser\_price']=disc

lists.append(prod\_inf)

value=[x for x in lists if x]

else:

productsQuerySet=Sup\_Upload1.objects(upload\_category=category,upload\_subcategory\_\_in=subcategory,upload\_brand\_\_in =brand,upload\_mrp\_\_gte=float(pricerange\_lw),upload\_mrp\_\_lte=float(pricerange\_hg)).only('salesqty','avgrating','upload\_name','upload\_id','upload\_brand','upload\_subcategory','upload\_mrp','end\_user\_disc').order\_by(filterStr)

print productsQuerySet,'sixth\_else'

if productsQuerySet.count()==0:

return {"status":"No data avialbale"}

lists=[]

products = json.loads(productsQuerySet.to\_json())[int(fromval):int(to)]

for prod in products:

print prod

print prod['upload\_id'],'upload\_id'

prod\_inf={}

total\_prod='011'+prod['upload\_id']+'.jpg'

total\_info=prod

for prods in files:

if prods==total\_prod:

print prods,'prods'

prod\_inf['product']=prod

prod\_inf['image']=prods

disc=float(prod['upload\_mrp'])-(float(prod['upload\_mrp'])\*float(prod['end\_user\_disc'])/100)

prod\_inf['enduser\_price']=disc

lists.append(prod\_inf)

value=[x for x in lists if x]

return {"status":"Success",'view':view,'filterdata':{'category':category,'subcategory':sub\_list,'brand':brands\_list,'sortdata':val,},"products":products,"from":fromval,"to":to,"totalcount":totalCagetoryProducts}

elif subcategory==[''] and brand==[''] and pricerange!='':

pricerange\_lw=pricerange.split('-')[0]

pricerange\_hg=pricerange.split('-')[1]

productsQuerySet=Sup\_Upload1.objects(upload\_category=category,upload\_mrp\_\_gte=float(pricerange\_lw),upload\_mrp\_\_lte=float(pricerange\_hg)).only('salesqty','avgrating','upload\_name','upload\_subcategory','upload\_id','upload\_mrp','end\_user\_disc').order\_by(filterStr)[int(fromval):int(to)]

print productsQuerySet,'seventh'

if productsQuerySet.count()==0:

return {"status":"No data avialbale"}

lists=[]

products = json.loads(productsQuerySet.to\_json())

for prod in products:

print prod

print prod['upload\_id'],'upload\_id'

prod\_inf={}

total\_prod='011'+prod['upload\_id']+'.jpg'

total\_info=prod

for prods in files:

if prods==total\_prod:

print prods,'prods'

prod\_inf['product']=prod

prod\_inf['image']=prods

disc=float(prod['upload\_mrp'])-(float(prod['upload\_mrp'])\*float(prod['end\_user\_disc'])/100)

prod\_inf['enduser\_price']=disc

lists.append(prod\_inf)

value=[x for x in lists if x]

subCategory = Sup\_Upload1.objects(upload\_category=category).only('upload\_subcategory').distinct('upload\_subcategory')

subcategories = json.loads(json.dumps(subCategory))

subCategories=sorted(subcategories)

brands = Sup\_Upload1.objects(upload\_category=category,).only('upload\_brand').distinct('upload\_brand')# it returns the list of brands, remove duplicates

branddata = json.loads(json.dumps(brands))

brandslist=sorted(branddata)

return {"status":"success","value":value,"from":fromval,"to":to,'filterdata':{'category':category,'subcategory':sub\_list,'brand':brands\_list,'pricerange':pricerange}}

elif subcategory!=[''] and brand==[''] and pricerange!='':

pricerange\_lw=pricerange.split('-')[0]

print pricerange\_lw

pricerange\_hg=pricerange.split('-')[1]

print pricerange\_hg

productsQuerySet=Sup\_Upload1.objects(upload\_subcategory\_\_in=subcategory,upload\_category=category,upload\_mrp\_\_gte=float(pricerange\_lw),upload\_mrp\_\_lte=float(pricerange\_hg)).only('avgrating','upload\_name','upload\_subcategory','upload\_id','salesqty','upload\_mrp','end\_user\_disc').order\_by(filterStr)[int(fromval):int(to)]

print productsQuerySet,'prevlast'

if productsQuerySet.count()==0:

return {"status":"No data avialbale"}

lists=[]

products = json.loads(productsQuerySet.to\_json())

#print products,'products'

for prod in products:

prod\_inf={}

total\_prod='011'+prod['upload\_id']+'.jpg'

total\_info=prod

for prods in files:

if prods==total\_prod:

print prods,'prods'

prod\_inf['product']=prod

prod\_inf['image']=prods

disc=float(prod['upload\_mrp'])-(float(prod['upload\_mrp'])\*float(prod['end\_user\_disc'])/100)

prod\_inf['enduser\_price']=disc

lists.append(prod\_inf)

value=[x for x in lists if x]

#print value,'value'

subCategory = Sup\_Upload1.objects(upload\_category=category,upload\_subcategory\_\_in=subcategory).only('upload\_subcategory').distinct('upload\_subcategory')

subcategories = json.loads(json.dumps(subCategory))

subCategories=sorted(subcategories)

brands = Sup\_Upload1.objects(upload\_category=category,upload\_subcategory\_\_in=subcategory,).only('upload\_brand').distinct('upload\_brand')# it returns the list of brands, remove duplicates

branddata = json.loads(json.dumps(brands))

brandslist=sorted(branddata)

return {"status":"success","value":value,"from":fromval,"to":to,'filterdata':{'category':category,'subcategory':sub\_list,'brand':brands\_list,'pricerange':pricerange}}

elif subcategory ==[''] and brand!=[''] and pricerange!='':

pricerange\_lw=pricerange.split('-')[0]

pricerange\_hg=pricerange.split('-')[1]

productsQuerySet=Sup\_Upload1.objects(upload\_brand\_\_in = brand,upload\_category=category,).only('avgrating','upload\_name','upload\_subcategory','upload\_id','salesqty','upload\_mrp','end\_user\_disc').order\_by(filterStr)[int(fromval):int(to)]

print productsQuerySet,'prevlast1'

if productsQuerySet.count()==0:

return {"status":"No data avialbale"}

lists=[]

products = json.loads(productsQuerySet.to\_json())

for prod in products:

print prod

print prod['upload\_id'],'upload\_id'

prod\_inf={}

total\_prod='011'+prod['upload\_id']+'.jpg'

total\_info=prod

for prods in files:

if prods==total\_prod:

print prods,'prods'

prod\_inf['product']=prod

prod\_inf['image']=prods

disc=float(prod['upload\_mrp'])-(float(prod['upload\_mrp'])\*float(prod['end\_user\_disc'])/100)

prod\_inf['enduser\_price']=disc

lists.append(prod\_inf)

value=[x for x in lists if x]

subCategory = Sup\_Upload1.objects(upload\_category=category,upload\_brand\_\_in = brand).only('upload\_subcategory').distinct('upload\_subcategory')

subcategories = json.loads(json.dumps(subCategory))

subCategories=sorted(subcategories)

brands = Sup\_Upload1.objects(upload\_category=category,upload\_brand\_\_in = brand).only('upload\_brand').distinct('upload\_brand')# it returns the list of brands, remove duplicates

branddata = json.loads(json.dumps(brands))

brandslist=sorted(branddata)

return {"status":"Success","value":value,'view':view,'filterdata':{'subcategory':sub\_list,'brand':brands\_list,'pricerange':pricerange,'sortdata':val},"from":fromval,"to":to}

class Categories(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

categories=Category.objects()

data=[]

for cat in categories:

sup=Sup\_Upload1.objects(upload\_category=cat.categoryname)

if sup.count()>0:

details={}

details['categoryname']=cat.categoryname

details['cat\_icon']=cat.cat\_icon

sub\_data=[]

for subcat in Sub\_Category.objects(categoryname=cat.categoryname):

sup=Sup\_Upload1.objects(upload\_subcategory=subcat.subcategory)

if sup.count()>0:

sub\_cat={}

sub\_cat['subcategory']=subcat.subcategory

json\_sub\_cat=json.loads(json.dumps(sub\_cat))

sub\_data.append(json\_sub\_cat)

sub\_cat=sorted(sub\_data)

details['subcategoryname']=sub\_cat

json\_cat=json.loads(json.dumps(details))

data.append(json\_cat)

return {"status":"success","categories":sorted(data)}

class Wishlist(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

cust\_email=CustomerDetails.objects.get(id=json\_data['user\_id'])

wish\_list=WishList.objects(user\_id=cust\_email.email)

if wish\_list.count()==0:

wish\_list=WishList(user\_id=cust\_email.email,product\_name=[json\_data['product\_name']])

wish\_list.save()

wish\_list=WishList.objects(user\_id=cust\_email.email)

items\_count=len(wish\_list[0].product\_name)

else:

wish\_list1=WishList.objects.get(user\_id=cust\_email.email)

if json\_data['product\_name'] in wish\_list1.product\_name:

return {"status":"product already existed in your wishlist"}

product\_append=wish\_list1.product\_name+[json\_data['product\_name']]

wish\_list1.product\_name=product\_append

wish\_list1.save()

items\_count=len(wish\_list1.product\_name)

return {"status":"product saved successfully","items\_count":items\_count}

class YourWishlist(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args=request.args

userid=args['user\_id']

files = os.listdir("./static/ProductImages")

cust\_email=CustomerDetails.objects.get(id=userid)

wishlist=WishList.objects(user\_id=cust\_email.email)

prod\_data=[]

for prods in wishlist:

for prod in prods.product\_name:

prod\_detail=Sup\_Upload1.objects(upload\_name=prod)

for product in prod\_detail:

data={}

data["upload\_name"]=product.upload\_name

data["description"]=product.description

data["upload\_name"]=product.upload\_name

data["upload\_mrp"]=product.upload\_mrp

data["percentage"]=product.end\_user\_disc

data["qty"]='1'

total\_prod='011'+product.upload\_id+'.jpg'

disc=float(data['upload\_mrp'])-(float(data['upload\_mrp'])\*float(data['percentage'])/100)

data['enduser\_price']=disc

for prods in files:

if prods==total\_prod:

data["image"]=prods

prod\_data.append(data)

return {"status":"success","prod\_info":prod\_data}

class RemoveWishlist(Resource):

def delete(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

userid=json\_data['user\_id']

prod\_name=json\_data['product\_name']

cust\_email=CustomerDetails.objects.get(id=userid)

wishlist=WishList.objects(user\_id=cust\_email.email)

if wishlist.count()==0:

return {"status":"user not registered"}

else:

wishlist1=WishList.objects.get(user\_id=cust\_email.email)

prod\_list=wishlist1.product\_name

for prod\_info in prod\_list:

if prod\_info == prod\_name:

wishlist1.product\_name.remove(prod\_info)

wishlist1.save()

wishlist\_count= len(wishlist1.product\_name)

print wishlist\_count

return {"status":"product removed successfully","wishlist\_count":wishlist\_count}

class YourWishlistCartCount(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args=request.args

userid=args['userid'] if 'userid' in args else ''

random\_no=args['random\_no'] if 'random\_no' in args else ''

if userid!='':

cart\_data=InitiateOrdersSite.objects.get(user\_id=args['userid'])

cart\_count=len(cart\_data.orderitemsite)

return {"status":"success","cart\_count":cart\_count}

if random\_no !='':

cart\_data=InitiateOrdersSite.objects.get(user\_id=random\_no)

cart\_count=len(cart\_data.orderitemsite)

return {"status":"success","cart\_count":cart\_count}

return {"status":"success"}

class SearchProduct(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args=request.args

prod\_name=args['product\_name']

if Sup\_Upload1.objects(upload\_name\_\_icontains=prod\_name).count()>0:

prod\_detail=Sup\_Upload1.objects(upload\_name\_\_icontains=prod\_name).only('upload\_name','upload\_brand','upload\_category','upload\_subcategory','upload\_modelno','upload\_netWeight','description',

'upload\_photo','avgrating','attributes\_\_atrname','attributes\_\_atrvalue','end\_user\_disc','upload\_mrp')

prod\_info=json.loads(prod\_detail.to\_json())

return {"status":"success","product\_info":prod\_info}

else:

return {"status":"fail","data":"No Data Available"}

class MatchProduct(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args=request.args

prod\_name=args['product\_name']

prod\_detail=Sup\_Upload1.objects(upload\_name\_\_icontains=prod\_name).only('upload\_name')

if prod\_detail.count()>0:

prod\_info=json.loads(prod\_detail.to\_json())

category\_detail=Sup\_Upload1.objects(upload\_name\_\_icontains=prod\_name).only('upload\_category')

category\_info=json.loads(category\_detail.to\_json())

categories=[dict(t) for t in set([tuple(d.items()) for d in category\_info])]

return {"status":"success","product\_info":prod\_info,"recommended":categories}

else:

return {"status":"fail","data":"No Products Matched For Entered keyword"}

class Compare\_Prod\_Details\_post(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

prod\_name=json\_data['product\_name']

files = os.listdir("./static/ProductImages")

prod\_data=[]

for prod in prod\_name:

prod\_detail=Sup\_Upload1.objects(upload\_name=prod)

for product in prod\_detail:

data={}

data["upload\_name"]=product.upload\_name

data["upload\_brand"]=product.upload\_brand

data["upload\_modelno"]=product.upload\_modelno

data["upload\_discount"]=product.upload\_discount

data["upload\_netWeight"]=product.upload\_netWeight

data["description"]=product.description

data["status"]=product.status

total\_prod='011'+product.upload\_id+'.jpg'

data["upload\_mrp"]=product.upload\_mrp

data["percentage"]=product.end\_user\_disc

disc=float(data['upload\_mrp'])-(float(data['upload\_mrp'])\*float(data['percentage'])/100)

data['enduser\_price']=disc

for prods in files:

if prods==total\_prod:

data["image"]=prods

for attr in product.attributes:

data['atrnam']=attr.atrname

data['atrvalue']=attr.atrvalue

prod\_data.append(data)

print len(prod\_data)

return {"status":"success","prod\_info":prod\_data}

class Add\_Compare\_Product(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

user\_id=json\_data['user\_id']

product=[json\_data['productname']]

cust\_details=CustomerDetails.objects.get(id=user\_id)

compare\_products=CompareProducts.objects(user\_id=cust\_details.email)

if compare\_products.count()==0:

compare\_prod=CompareProducts(user\_id=cust\_details.email,product\_name=product)

compare\_prod.save()

else:

compare\_prod=CompareProducts.objects.get(user\_id=cust\_details.email)

if json\_data['productname'] in compare\_prod.product\_name:

return {"status":"product already existed"}

product\_append=compare\_prod.product\_name+product

compare\_prod.product\_name=product\_append

compare\_prod.save()

print "saving"

return {"status":"product saved successfully","product\_info":compare\_products[0].product\_name}

class Comp\_Prod\_Details\_get(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args=request.args

user\_id=args['user\_id']

cust\_details=CustomerDetails.objects.get(id=user\_id)

compare\_products=CompareProducts.objects(user\_id=cust\_details.email)

if compare\_products.count()>0:

compare\_product\_info=CompareProducts.objects.get(user\_id=cust\_details.email)

print compare\_product\_info,'display'

return {"status":"success","prod\_info":compare\_product\_info.product\_name}

else:

return {"status":"success","prod\_info":""}

class Comp\_Prod\_Delete(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args=request.args

json\_data=request.get\_json(force=True)

prod\_name=json\_data['product\_name']

user\_id=json\_data['user\_id']

compare\_products=args['compare\_products']

if compare\_products=='':

cust\_details=CustomerDetails.objects.get(id=user\_id)

compare\_products=CompareProducts.objects.get(user\_id=cust\_details.email)

compare\_products.product\_name.remove(prod\_name)

compare\_products.save()

comp\_prod=CompareProducts.objects.get(user\_id=cust\_details.email)

return {"status":"success","prod\_info":comp\_prod.product\_name}

else:

cust\_details=CustomerDetails.objects.get(id=user\_id)

compare\_products=CompareProducts.objects.get(user\_id=cust\_details.email)

compare\_products.delete()

return {"status":"success"}

class Your\_Orders(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args=request.args

userid=args['user\_id']

cust\_email=CustomerDetails.objects.get(id=userid)

cust\_orders=CustomerOrders.objects(customermobile=cust\_email.mobile)

if cust\_orders.count()>0:

if cust\_orders[0].paymenttype=="cashondelivery":

cust\_orders=CustomerOrders.objects(customermobile=cust\_email.mobile,payment\_status='cash').only('orderid','createddate','totalquantity','totalamount','status').order\_by("-id")

order\_info=json.loads(cust\_orders.to\_json())

return {"status":"success","order\_info":order\_info}

else:

cust\_orders=CustomerOrders.objects(customermobile=cust\_email.mobile,payment\_status='yes').only('orderid','createddate','totalquantity','totalamount','status').order\_by("-id")

order\_info=json.loads(cust\_orders.to\_json())

return {"status":"success","order\_info":order\_info}

else:

return {"status":"success","order\_info":""}

class Pending\_Orders(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args=request.args

userid=args['user\_id']

cust\_email=CustomerDetails.objects.get(id=userid)

cust\_orders=CustomerOrders.objects(customermobile=cust\_email.mobile,status='Accepted').only('orderid','createddate','totalquantity','totalamount','status').order\_by("-id")

order\_info=json.loads(cust\_orders.to\_json())

return {"status":"success","user\_info":order\_info}

class Invoices(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args=request.args

userid=args['user\_id']

cust\_email=CustomerDetails.objects.get(id=userid)

print cust\_email.mobile

confirm\_inv=ConfirmInvoice.objects(user\_mobile=cust\_email.mobile).order\_by("-id")

invoice\_info=[]

for cust in confirm\_inv:

invoices={}

invoices['orderid']=cust.orderid

invoices['invoiceid']=cust.invoiceid

invoices['paymenttype']=cust.paymenttype

invoices['totalamount']=cust.totalamount

cust\_orders=CustomerOrders.objects.get(orderid=cust.orderid)

for items in cust\_orders.orderitems:

invoices['productdescription']=items.productdescription

invoices['offer\_price']=items.unitprice

json\_invoice=json.loads(json.dumps(invoices))

invoice\_info.append(json\_invoice)

return {"status":"success","user\_info":invoice\_info}

class Order\_Data(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args=request.args

orderid=args['orderid']

total=0

order\_data=[]

for cust\_orders in CustomerOrders.objects(orderid=orderid):

cust\_details={}

cust\_det=CustomerDetails.objects.get(mobile=cust\_orders.customermobile)

cust\_details['gst\_number']=cust\_det.gst\_number

cust\_details['paymenttype']=cust\_orders.paymenttype

cust\_details['shippingtype']=cust\_orders.shippingtype

cust\_details['status']=cust\_orders.status

cust\_details['discount\_amount']=cust\_orders.discountAmount

cust\_details['shop']=cust\_orders.shop

grand\_total=cust\_orders.totalamount

for ship\_adr in cust\_orders.shippingaddress:

ship\_adrs={"firstname":ship\_adr.firstname,"lastname":ship\_adr.lastname,"house\_no":ship\_adr.house\_no,"street\_address":ship\_adr.street\_address,"city":ship\_adr.city,"state":ship\_adr.state,"postal\_code":ship\_adr.postal\_code,"country":ship\_adr.country,"mobile":ship\_adr.mobile[2:],"address\_id":ship\_adr.address\_id}

cust\_details['shippingaddress']=ship\_adrs

for bill\_adr in cust\_orders.billingaddress:

bill\_adrs={"firstname":bill\_adr.firstname,"lastname":bill\_adr.lastname,"house\_no":bill\_adr.house\_no,"street\_address":bill\_adr.street\_address,"city":bill\_adr.city,"state":bill\_adr.state,"postal\_code":bill\_adr.postal\_code,"country":bill\_adr.country,"mobile":bill\_adr.mobile[2:],"address\_id":bill\_adr.address\_id}

cust\_details['billingaddress']=bill\_adrs

if cust\_orders.shippingtype=="Pickup":

shop\_setup=Shopsetup.objects(shop\_name=cust\_orders.shop)

if shop\_setup.count == 0:

return {"status":" shop name invalid"}

else:

for addr in shop\_setup:

cust\_details['pickup\_address']=addr.shop\_address

data=json.loads(json.dumps(cust\_details))

for order in cust\_orders.orderitems:

order\_items={}

order\_items['productdescription']=order.productdescription

Sup\_Upload1=Sup\_Upload1.objects(upload\_name=order.productdescription)

for prod in Sup\_Upload1:

order\_items['avgrating']=prod.avgrating

order\_items['unitprice']=order.unitprice

order\_items['tax']=order.tax

order\_items['tax\_amount']=order.tax\_amount

tax=order.tax

order\_items['qty']=order.qty

tax\_amount=round((float(order\_items['unitprice'])\*(float(tax)/100))\*float(order\_items['qty']),2)

total+=tax\_amount

order\_items['sub\_total']=order.sub\_total

json\_orderitems=json.loads(json.dumps(order\_items))

order\_data.append(json\_orderitems)

return {"status":"success","user\_info":{"cust\_details":data, "order\_data":order\_data,"grand\_total":grand\_total,"tax\_amount":total}}

class Cancel\_Order(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

#pool = Pool()

args=request.args

orderid=args['orderid']

cust\_orders=CustomerOrders.objects.get(orderid=orderid)

cust\_orders.status='Cancel'

cust\_orders.save()

cust\_details=CustomerDetails.objects.get(mobile=cust\_orders.customermobile)

mobile=cust\_orders.customermobile

msg="Dear "+cust\_details.firstname+cust\_details.lastname+' , '+"your order "+orderid+" is cancelled."

SMS(msg,mobile)

#pool.apply\_async(SMS,[msg,mobile])

fo = open("./templates/cancel\_order.html", "r+")

htmlbody = fo.read()

fo.close()

urldata="Dear "+cust\_details.firstname+cust\_details.lastname+' , '+" your order "+orderid+" is cancelled."

htmlbody = htmlbody.replace("$$urldata$$",urldata)

#pool.apply\_async(sendMail,[cust\_details.email,'ShopMyTools',htmlbody])

sendMail(cust\_details.email,'Order Cancel',htmlbody)

shop=cust\_orders.shop

shop\_user=ShopUser.objects(shop=shop)

mobile=shop\_user[0].mobile

msg="Dear "+shop\_user[0].username+' , '+orderid+" is cancelled."

SMS(msg,mobile)

fo = open("./templates/cancel\_order.html", "r+")

htmlbody = fo.read()

fo.close()

urldata="Dear "+shop\_user[0].username+' , '+orderid+" is cancelled."

htmlbody = htmlbody.replace("$$urldata$$",urldata)

sendMail(shop\_user[0].email,'Order Cancel',htmlbody)

if cust\_orders.paymenttype=="payu" and payment\_status=="yes":

mobile='9100115016'

msg="Dear "+cust\_details.firstname+cust\_details.lastname+' , '+orderid+" is cancelled."

SMS(msg,mobile)

fo = open("./templates/cancel\_order.html", "r+")

htmlbody = fo.read()

fo.close()

urldata="Dear "+cust\_details.firstname+cust\_details.lastname+' , '+orderid+" is cancelled."

htmlbody = htmlbody.replace("$$urldata$$",urldata)

sendMail('usha@goldensuntechnology.com','Order Cancel',htmlbody)

return {"status":"success"}

#---------smtcheackout------------------

class InitiateOrder(Resource):

def post(self):

json\_data = request.get\_json(force=True)

files = os.listdir("./static/ProductImages")

user\_id=json\_data['user\_id']

req\_randomnumber=json\_data['random\_no'] if 'random\_no' in json\_data else ''

if json\_data['user\_id']=='':

init\_order=InitiateOrders.objects(user\_id=req\_randomnumber)

if init\_order.count() == 0:#if not random num

listofno = random.sample(set([1,2,3,4,5,6,7,8,9,'a','b','c','d','e','f','g']),16)

randomnumber="".join(str(numbers) for numbers in listofno)

customer\_order=InitiateOrders(user\_id=randomnumber)

totalprice=0.0

for order\_items in orderitems\_list:

brand=order\_items['brand']

discount=order\_items['discount']

imageid=order\_items['image']

mrp=order\_items['mrp']

product\_id=order\_items['product\_id']

saving\_percentage=order\_items['saving\_percentage']

spl\_discount=order\_items['special\_discount']

total=order\_items['total']

upload\_modelno=order\_items['upload\_modelno']

upload\_name=order\_items['upload\_name']

volume\_discount=order\_items['volume\_discount']

qty=str(order\_items['qty'])

supp\_product=Sup\_Upload1.objects(upload\_name=upload\_name)

totalprice+=flaot(total)

order\_details=Orderitems(upload\_name,str(qty),upload\_modelno,str(volume\_discount),str(saving\_percentage),str(spl\_discount),imageid,product\_id,brand,str(discount),str(total),str(mrp))

customer\_order.orderitem.append(order\_details)

customer\_order.save()

total\_price=format(totalprice,'.2f')

return {"status":"item added to cart","random\_number":randomnumber,"total\_price":total\_price}

else:#if random num

order\_doc=InitiateOrders.objects.get(user\_id=req\_randomnumber)

order\_doc.orderitem=[]

order\_doc.save()

totalprice=0.0

for order\_items in final\_list:

brand=order\_items['brand']

discount=order\_items['discount']

imageid=order\_items['image']

mrp=order\_items['mrp']

product\_id=order\_items['product\_id']

saving\_percentage=order\_items['saving\_percentage']

spl\_discount=order\_items['special\_discount']

total=order\_items['total']

upload\_modelno=order\_items['upload\_modelno']

upload\_name=order\_items['upload\_name']

volume\_discount=order\_items['volume\_discount']

qty=str(order\_items['qty'])

totalprice+=float(total)

supp\_product=Sup\_Upload1.objects(upload\_name=upload\_name)

order\_details=Orderitems(upload\_name,str(qty),upload\_modelno,str(volume\_discount),str(saving\_percentage),str(spl\_discount),imageid,product\_id,brand,str(discount),str(total),str(mrp))

order\_doc.orderitem.append(order\_details)

order\_doc.save()

total\_price=format(totalprice,'.2f')

return {"status":"item added to cart","total\_price":total\_price}

elif json\_data['user\_id'] != '': #if userid something

order\_status=json\_data['order\_status']

if order\_status=='init':

check\_cart=InitiateOrders.objects(user\_id=user\_id)

if check\_cart.count() == 0:

print "initial"

customer\_order=InitiateOrders(user\_id=user\_id)

totalprice=0.0

for order\_items in json\_data['orderitem']:

discount=order\_items['discount']

brand=order\_items['brand']

imageid=order\_items['image']

mrp=order\_items['mrp']

product\_id=order\_items['product\_id']

saving\_percentage=order\_items['saving\_percentage']

spl\_discount=order\_items['special\_discount']

total=order\_items['total']

upload\_modelno=order\_items['upload\_modelno']

upload\_name=order\_items['upload\_name']

volume\_discount=order\_items['volume\_discount']

qty=str(order\_items['qty'])

totalprice+=float(total)

supp\_product=Sup\_Upload1.objects(upload\_name=upload\_name)

if supp\_product.count()>0:

order\_details=Orderitems(upload\_name,str(qty),upload\_modelno,str(volume\_discount),str(saving\_percentage),str(spl\_discount),imageid,product\_id,str(brand),str(discount),str(total),str(mrp))

customer\_order.orderitem.append(order\_details)

customer\_order.save()

cart\_count=InitiateOrders.objects.get(user\_id=user\_id)

cart\_data= len(cart\_count.orderitem)

total\_price=format(totalprice,'.2f')

return {"status":"item added to cart","cart\_count":cart\_data,"total\_price":total\_price}

if check\_cart.count() > 0:

for order\_items in json\_data['orderitem']:

brand=order\_items['brand']

print type(brand)

discount=order\_items['discount']

imageid=order\_items['image']

mrp=order\_items['mrp']

product\_id=order\_items['product\_id']

saving\_percentage=order\_items['saving\_percentage']

spl\_discount=order\_items['special\_discount']

total=order\_items['total']

upload\_modelno=order\_items['upload\_modelno']

upload\_name=order\_items['upload\_name']

volume\_discount=order\_items['volume\_discount']

qty=str(order\_items['qty'])

supp\_product=Sup\_Upload1.objects(upload\_name=upload\_name)

if supp\_product.count()>0:

order\_doc1=InitiateOrders.objects(user\_id=user\_id).filter(orderitem\_\_match={'productname':upload\_name})

if order\_doc1:

print "hi"

order\_doc=InitiateOrders.objects.get(user\_id=user\_id)

for prod in order\_doc.orderitem:

if prod.productname==upload\_name:

qty1=int(qty)+int(prod.qty)

total1=float(total)+float(prod.total)

spcl\_disc=float(spl\_discount)+float(prod.spl\_discount)

vol\_disc=float(volume\_discount)+float(prod.volume\_discount)

prod.qty=str(qty1)

prod.total=str(total1)

prod.spl\_discount=str(spcl\_disc)

prod.volume\_discount=str(vol\_disc)

prod.save()

order\_doc1=InitiateOrders.objects.get(user\_id=user\_id)

totalprice=sum(float(prods['total']) for prods in order\_doc1.orderitem)

total\_price=format(totalprice,'.2f')

else:

print "else"

order\_doc2=InitiateOrders.objects.get(user\_id=user\_id)

order\_details=Orderitems(upload\_name,str(qty),upload\_modelno,str(volume\_discount),str(saving\_percentage),str(spl\_discount),imageid,product\_id,str(brand),str(discount),str(total),str(mrp))

order\_doc2.orderitem.append(order\_details)

order\_doc2.save()

order\_doc=InitiateOrders.objects.get(user\_id=user\_id)

totalprice=sum(float(prod['total']) for prod in order\_doc.orderitem)

total\_price=format(totalprice,'.2f')

cart\_count=InitiateOrders.objects.get(user\_id=user\_id)

cart\_data= len(cart\_count.orderitem)

return {"status":"item added to cart","cart\_count":cart\_data,"total\_price":total\_price}

else:

return {"status":"invalid status"}

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args = request.args

userid=args['userid'] if 'userid' in args else ''

random\_no=args['random\_no'] if 'random\_no' in args else ''

#data1=InitiateOrders.objects.get(user\_id=random\_no)

if userid != '':

rand\_order\_detail=InitiateOrders.objects(user\_id=userid)

if rand\_order\_detail.count()==0:

return {"status":"cart is empyt."}

total\_item\_list=[]

rand\_order\_details=InitiateOrders.objects.get(user\_id=userid)

user\_id=rand\_order\_details.user\_id

for item in rand\_order\_details.orderitem:

data={}

data['upload\_name']=item.productname

data['qty']=item.qty

data['upload\_modelno']=item.modelno

data['volume\_discount']=item.volume\_discount

data['saving\_percentage']=item.saving\_percentage

data['special\_discount']=item.spl\_discount

data['imageid']=item.imageid

data['product\_id']=item.product\_id

data['brand']=item.brand

data['discount']=item.discount

data['total']=item.total

data['mrp']=item.mrp

supp\_product=Sup\_Upload1.objects.get(upload\_name=item.productname)

attr\_list=[]

for attr in supp\_product.attributes:

attr\_data={"atrname":attr.atrname,"atrvalue":attr.atrvalue,"tolerance1":attr.tolerance1,"tolerance2":attr.tolerance2}

attr\_list.append(attr\_data)

data['attributes']=attr\_list

data['pieceperCarton']=supp\_product.upload\_pieceperCarton

data['standard\_pack']=supp\_product.STD\_non\_pack

json\_data=json.loads(json.dumps(data))

total\_item\_list.append(json\_data)

totalprice=sum(float(prod['total']) for prod in rand\_order\_details.orderitem)

total\_price=format(totalprice,'.2f')

return {"status":"Success","user\_id":userid,"total\_item\_list":total\_item\_list,"total\_price":total\_price}

if random\_no !='':

total\_item\_list=[]

rand\_order\_details=InitiateOrders.objects.get(user\_id=random\_no)

user\_id=rand\_order\_details.user\_id

for item in rand\_order\_details.orderitem:

data={}

data['upload\_name']=item.productname

data['qty']=item.qty

data['upload\_modelno']=item.modelno

data['volume\_discount']=item.volume\_discount

data['saving\_percentage']=item.saving\_percentage

data['special\_discount']=item.spl\_discount

data['imageid']=item.imageid

data['product\_id']=item.product\_id

data['brand']=item.brand

data['discount']=item.discount

data['total']=item.total

data['mrp']=item.mrp

supp\_product=Sup\_Upload1.objects.get(upload\_name=item.productname)

attr\_list=[]

for attr in supp\_product.attributes:

attr\_data={"atrname":attr.atrname,"atrvalue":attr.atrvalue,"tolerance1":attr.tolerance1,"tolerance2":attr.tolerance2}

attr\_list.append(attr\_data)

data['attributes']=attr\_list

data['pieceperCarton']=supp\_product.upload\_pieceperCarton

data['standard\_pack']=supp\_product.standard\_pack

json\_data=json.loads(json.dumps(data))

total\_item\_list.append(json\_data)

totalprice=sum(float(prod['total']) for prod in rand\_order\_details.orderitem)

total\_price=format(totalprice,'.2f')

return {"status":"Success","user\_id":user\_id,"total\_item\_list":total\_item\_list,"total\_price":total\_price}

def delete(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args = request.args

json\_data = request.get\_json(force=True)

userid=args['userid']

product=json\_data['product']

if product == 'all':

order\_del=InitiateOrders.objects.get(user\_id=userid)

order\_del.delete()

return {"status":"all products deleted successfully"}

cart\_item = InitiateOrders.objects.get(user\_id=userid)

a=[i for i,j in enumerate(cart\_item.orderitem) if j['productname'] == json\_data['product']][0]

print a

del cart\_item.orderitem[a]

cart\_item.save()

if len(cart\_item.orderitem)==0:

order\_del=InitiateOrders.objects.get(user\_id=userid)

order\_del.delete()

return {"status":"product deleted successfully"}

else:

return {"status":"invalid secret key"}

def put(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

userid=json\_data['user\_id']

for order\_items in json\_data['orderitem']:

discount=str(order\_items['discount'])

brand=str(order\_items['brand'])

imageid=str(order\_items['imageid'])

mrp=str(order\_items['mrp'])

product\_id=str(order\_items['product\_id'])

saving\_percentage=str(order\_items['saving\_percentage'])

spl\_discount=str(order\_items['special\_discount'])

total=str(order\_items['total'])

upload\_modelno=str(order\_items['upload\_modelno'])

upload\_name=str(order\_items['upload\_name'])

volume\_discount=str(order\_items['volume\_discount'])

qty=str(order\_items['qty'])

rand\_order\_details=InitiateOrders.objects.get(user\_id=userid)

for item in rand\_order\_details.orderitem:

if item.productname==upload\_name:

item.productname=upload\_name

item.qty=qty

item.modelno=upload\_modelno

item.volume\_discount=volume\_discount

item.saving\_percentage=saving\_percentage

item.spl\_discount=spl\_discount

item.imageid=imageid

item.product\_id=product\_id

item.brand=brand

item.discount=discount

item.total=total

item.mrp=mrp

item.save()

rand\_order\_details=InitiateOrders.objects.get(user\_id=userid)

totalprice=sum(float(prod['total']) for prod in rand\_order\_details.orderitem)

total\_price=format(totalprice,'.2f')

order\_count=len(rand\_order\_details.orderitem)

return {"status":"success","updating\_data":"details updated successfully.","total\_price":total\_price,"cart\_count":order\_count}

else:

return {"status":"invalid secret key"}

class AltrProducts(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

files = os.listdir("./static/ProductImages")

prod\_id = request.args['product\_id']

proddata=Sup\_Upload1.objects(upload\_id=prod\_id)

prod\_data=[]

for prod in proddata:

for up\_id in prod.altr\_upload\_id:

data=Sup\_Upload1.objects(upload\_id=up\_id)

for prod\_info in data:

prod\_details={}

prod\_details['upload\_name']=prod\_info.upload\_name

prod\_details['upload\_category']=prod\_info.upload\_category

prod\_details['upload\_subcategory']=prod\_info.upload\_subcategory

prod\_details["upload\_mrp"]=prod\_info.upload\_mrp

prod\_details["percentage"]=prod\_info.end\_user\_disc

disc=float(prod\_details['upload\_mrp'])-(float(prod\_details['upload\_mrp'])\*float(prod\_details['percentage'])/100)

prod\_details['enduser\_price']=disc

total\_prod='011'+prod\_info.upload\_id+'.jpg'

for prods in files:

if prods==total\_prod:

prod\_details['image']=prods

print prod\_details['image']

attributes=[]

for j in prod\_info.attributes:

details1={}

details1['atrname']=j.atrname

details1['atrvalue']=j.atrvalue

test=json.loads(json.dumps(details1))

attributes.append(test)

prod\_details['attributes']=attributese

prod\_data.append(prod\_details)

return {'status':'Success',"offer\_products":prod\_data}

class SparProducts(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

files = os.listdir("./static/ProductImages")

prod\_id = request.args['product\_id']

proddata=Sup\_Upload1.objects(upload\_id=prod\_id)

prod\_data=[]

for prod in proddata:

for up\_id in prod.spar\_upload\_id:

data=Sup\_Upload1.objects(upload\_id=up\_id)

for prod\_info in data:

prod\_details={}

prod\_details['upload\_name']=prod\_info.upload\_name

prod\_details['upload\_category']=prod\_info.upload\_category

prod\_details['upload\_subcategory']=prod\_info.upload\_subcategory

prod\_details["upload\_mrp"]=prod\_info.upload\_mrp

prod\_details["percentage"]=prod\_info.end\_user\_disc

disc=float(prod\_details['upload\_mrp'])-(float(prod\_details['upload\_mrp'])\*float(prod\_details['percentage'])/100)

prod\_details['enduser\_price']=disc

total\_prod='011'+prod\_info.upload\_id+'.jpg'

for prods in files:

if prods==total\_prod:

prod\_details['image']=prods

print prod\_details['image']

attributes=[]

for j in prod\_info.attributes:

details1={}

details1['atrname']=j.atrname

details1['atrvalue']=j.atrvalue

test=json.loads(json.dumps(details1))

attributes.append(test)

prod\_details['attributes']=attributes

prod\_data.append(prod\_details)

return {'status':'Success',"offer\_products":prod\_data}

class CatSubBrand(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

categories=Category.objects()

subcategories=Sub\_Category.objects()

cat\_data=[]

for cat in categories:

if Sup\_Upload1.objects(upload\_category=cat.categoryname).count()>0:

cat\_details={}

cat\_details["categoryname"]=cat.categoryname

cat\_data.append(cat\_details)

categories=[dict(t) for t in set([tuple(d.items()) for d in cat\_data])]

subcat\_data=[]

for subcat in subcategories:

if Sup\_Upload1.objects(upload\_subcategory=subcat.subcategory).count()>0:

subcat\_details={}

subcat\_details["subcategory"]=subcat.subcategory

subcat\_data.append(subcat\_details)

subcategories=[dict(t) for t in set([tuple(d.items()) for d in subcat\_data])]

Sup=Sup\_Upload1.objects()

brand\_data=[]

for brand in Sup:

brand\_details={}

brand\_details['brand']=brand.upload\_brand

brand\_data.append(brand\_details)

brands=[dict(t) for t in set([tuple(d.items()) for d in brand\_data])]

return {'status':'Success',"categories":sorted(categories),"subcategories":sorted(subcategories),"brands":sorted(brands)}

class SubcategoryData(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

brand\_data=[]

subcat\_data=[]

for cat in json\_data['category']:

print cat

for subcat in Sub\_Category.objects(categoryname=cat):

if Sup\_Upload1.objects(upload\_subcategory=subcat.subcategory).count()>0:

sub\_cat\_details={}

sub\_cat\_details['subcategory']=subcat.subcategory

subcat\_data.append(sub\_cat\_details)

subcategories=[dict(t) for t in set([tuple(d.items()) for d in subcat\_data])]

Sup=Sup\_Upload1.objects(upload\_category=cat)

for brand in Sup:

brand\_details={}

brand\_details['brand']=brand.upload\_brand

brand\_data.append(brand\_details)

brands=[dict(t) for t in set([tuple(d.items()) for d in brand\_data])]

return {'status':'Success',"subcategories":sorted(subcategories),"brands":sorted(brands)}

class Sub\_Brand(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

brand\_data=[]

for subcat in json\_data['subcategory']:

for sub\_cat in Sub\_Category.objects(subcategory=subcat):

Sup=Sup\_Upload1.objects(upload\_subcategory=sub\_cat.subcategory)

for brand in Sup:

brand\_details={}

brand\_details['brand']=brand.upload\_brand

brand\_data.append(brand\_details)

brands=[dict(t) for t in set([tuple(d.items()) for d in brand\_data])]

return {'status':'Success',"brands":sorted(brands)}

class CatSubcatBrandGet(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

files = os.listdir("./static/ProductImages")

json\_data=request.get\_json(force=True)

category = json\_data['category']

sub\_category = json\_data['sub\_category']

brand = json\_data['brand']

cat\_list=list(set(category))

sub\_list=list(set(sub\_category))

brands\_list=list(set(brand))

print len(cat\_list)

if len(cat\_list)==0 and len(sub\_list)==0 and len(brands\_list)==0:

product\_info=[]

proddata=Sup\_Upload1.objects(upload\_category="Power Tools")

for prod in proddata:

details={}

details['upload\_name']=prod.upload\_name

details['upload\_modelno']=prod.upload\_modelno

details['mrp']=prod.upload\_mrp

details['discount']=prod.upload\_discount

details['brand']=prod.upload\_brand

details['qty']='0'

details['product\_id']=prod.upload\_id

details['pieceperCarton']=prod.upload\_pieceperCarton

details['standard\_pack']=prod.STD\_non\_pack

details['upload\_mrp']=prod.upload\_mrp

details["percentage"]=prod.end\_user\_disc

disc=float(details['upload\_mrp'])-(float(details['upload\_mrp'])\*float(details['percentage'])/100)

details['enduser\_price']=disc

total\_prod='011'+prod.upload\_id+'.jpg'

attr\_list=[]

for attr in prod.attributes:

attributes={}

attributes['atrname']=attr.atrname

attributes['atrvalue']=attr.atrvalue

attr\_list.append(attributes)

details['attributes']=attr\_list

'''

price\_list=[]

for prc in prod.prices:

price={}

price['offer\_price']=prc.offer\_price

price['enduser\_price']=prc.enduser\_price

price\_list.append(price)

details['priceinfo']=price\_list

'''

spcl\_offer=ProductOffer.objects(productid=prod.upload\_id)

if spcl\_offer.count()>0:

details['special\_discount']=spcl\_offer[0].discount

total\_prod='011'+prod.upload\_id+'.jpg'

for prods in files:

if prods==total\_prod:

details['image']=prods

product\_info.append(details)

return {'status':'Success',"product\_data":product\_info}

if len(cat\_list)>0 and len(sub\_list)==0 and len(brands\_list)==0:

product\_info=[]

for cat in cat\_list:

proddata=Sup\_Upload1.objects(upload\_category=cat,status='Accept')

#product\_info=[]

for prod in proddata:

details={}

details['upload\_name']=prod.upload\_name

details['upload\_modelno']=prod.upload\_modelno

details['mrp']=prod.upload\_mrp

details['discount']=prod.upload\_discount

details['brand']=prod.upload\_brand

details['qty']=prod.salesqty

details['product\_id']=prod.upload\_id

details['pieceperCarton']=prod.upload\_pieceperCarton

details['standard\_pack']=prod.standard\_pack

details['upload\_mrp']=prod.upload\_mrp

details["percentage"]=prod.end\_user\_disc

disc=float(details['upload\_mrp'])-(float(details['upload\_mrp'])\*float(details['percentage'])/100)

details['enduser\_price']=disc

attr\_list=[]

for attr in prod.attributes:

attributes={}

attributes['atrname']=attr.atrname

attributes['atrvalue']=attr.atrvalue

attr\_list.append(attributes)

details['attributes']=attr\_list

spcl\_offer=ProductOffer.objects(productid=prod.upload\_id)

if spcl\_offer.count()>0:

details['special\_discount']=spcl\_offer[0].discount

total\_prod='011'+prod.upload\_id+'.jpg'

for prods in files:

if prods==total\_prod:

details['image']=prods

product\_info.append(details)

return {'status':'Success',"product\_data":product\_info}

elif len(cat\_list)>0 and len(sub\_list)>0 and len(brands\_list)==0:

print 'helllooooooooo'

product\_info=[]

for cat in cat\_list:

for subcat in sub\_list:

proddata=Sup\_Upload1.objects(upload\_category=cat,upload\_subcategory=subcat)

#product\_info=[]

for prod in proddata:

details={}

details['upload\_name']=prod.upload\_name

details['upload\_modelno']=prod.upload\_modelno

details['mrp']=prod.upload\_mrp

details['discount']=prod.upload\_discount

details['brand']=prod.upload\_brand

details['qty']=0

details['product\_id']=prod.upload\_id

details['pieceperCarton']=prod.upload\_pieceperCarton

details['standard\_pack']=prod.standard\_pack

details['upload\_mrp']=prod.upload\_mrp

details["percentage"]=prod.end\_user\_disc

disc=float(details['upload\_mrp'])-(float(details['upload\_mrp'])\*float(details['percentage'])/100)

details['enduser\_price']=disc

spcl\_offer=ProductOffer.objects(productid=prod.upload\_id)

if spcl\_offer.count()>0:

details['special\_discount']=spcl\_offer[0].discount

total\_prod='011'+prod.upload\_id+'.jpg'

for prods in files:

if prods==total\_prod:

details['image']=prods

product\_info.append(details)

print product\_info

return {'status':'Success',"product\_data":product\_info}

elif len(cat\_list)>0 and len(sub\_list)>0 and len(brands\_list)>0:

sub\_objects\_list=[]

for cat in cat\_list:

for subcat in sub\_list:

for sub\_obj in Sub\_Category.objects(subcategory=subcat,categoryname=cat):

sub\_objects=json.loads(sub\_obj.to\_json())

sub\_objects\_list.append(sub\_objects)

seq=[x['subcategory'] for x in sub\_objects\_list]

product\_info=[]

for sub in seq:

for brand in brands\_list:

proddata=Sup\_Upload1.objects(upload\_subcategory=sub,upload\_brand=brand)

for prod in proddata:

details={}

details['upload\_name']=prod.upload\_name

details['upload\_modelno']=prod.upload\_modelno

details['mrp']=prod.upload\_mrp

details['discount']=prod.upload\_discount

details['brand']=prod.upload\_brand

#details['qty']=prod.salesqty

details['qty']=0

details['product\_id']=prod.upload\_id

details['pieceperCarton']=prod.upload\_pieceperCarton

details['standard\_pack']=prod.standard\_pack

details['total']=0

details['saving\_percentage']=0

details['volume\_discount']=0

attr\_list=[]

for attr in prod.attributes:

attributes={}

attributes['atrname']=attr.atrname

attributes['atrvalue']=attr.atrvalue

attr\_list.append(attributes)

details['attributes']=attr\_list

details['upload\_mrp']=prod.upload\_mrp

details["percentage"]=prod.end\_user\_disc

disc=float(details['upload\_mrp'])-(float(details['upload\_mrp'])\*float(details['percentage'])/100)

details['enduser\_price']=disc

spcl\_offer=ProductOffer.objects(productid=prod.upload\_id)

if spcl\_offer.count()>0:

details['special\_discount']=spcl\_offer[0].discount

if spcl\_offer.count()==0:

details['special\_discount']=0

total\_prod='011'+prod.upload\_id+'.jpg'

for prods in files:

if prods==total\_prod:

details['image']=prods

product\_info.append(details)

return {'status':'Success',"product\_data":product\_info}#,"total"etails['total']}

elif len(cat\_list)>0 and len(sub\_list)==0 and len(brands\_list)>0:

product\_info=[]

for cat in cat\_list:

for brand in brands\_list:

proddata=Sup\_Upload1.objects(upload\_category=cat,upload\_brand=brand)

for prod in proddata:

details={}

details['upload\_name']=prod.upload\_name

details['upload\_modelno']=prod.upload\_modelno

details['mrp']=prod.upload\_mrp

details['discount']=prod.upload\_discount

details['brand']=prod.upload\_brand

details['qty']=0

details['product\_id']=prod.upload\_id

details['pieceperCarton']=prod.upload\_pieceperCarton

details['standard\_pack']=prod.standard\_pack

details['upload\_mrp']=prod.upload\_mrp

details["percentage"]=prod.end\_user\_disc

disc=float(details['upload\_mrp'])-(float(details['upload\_mrp'])\*float(details['percentage'])/100)

details['enduser\_price']=disc

spcl\_offer=ProductOffer.objects(productid=prod.upload\_id)

if spcl\_offer.count()>0:

details['special\_discount']=spcl\_offer[0].discount

total\_prod='011'+prod.upload\_id+'.jpg'

for prods in files:

if prods==total\_prod:

details['image']=prods

product\_info.append(details)

return {'status':'Success',"product\_data":product\_info}

elif len(cat\_list)==0 and len(sub\_list)==0 and len(brands\_list)>0:

product\_info=[]

for brand in brands\_list:

proddata=Sup\_Upload1.objects(upload\_brand=brand)

for prod in proddata:

details={}

details['upload\_name']=prod.upload\_name

details['upload\_modelno']=prod.upload\_modelno

details['mrp']=prod.upload\_mrp

details['discount']=prod.upload\_discount

details['brand']=prod.upload\_brand

details['qty']=prod.salesqty

details['product\_id']=prod.upload\_id

details['pieceperCarton']=prod.upload\_pieceperCarton

details['standard\_pack']=prod.standard\_pack

details['upload\_mrp']=prod.upload\_mrp

details["percentage"]=prod.end\_user\_disc

disc=float(details['upload\_mrp'])-(float(details['upload\_mrp'])\*float(details['percentage'])/100)

details['enduser\_price']=disc

spcl\_offer=ProductOffer.objects(productid=prod.upload\_id)

if spcl\_offer.count()>0:

details['special\_discount']=spcl\_offer[0].discount

total\_prod='011'+prod.upload\_id+'.jpg'

for prods in files:

if prods==total\_prod:

details['image']=prods

product\_info.append(details)

return {'status':'Success',"product\_data":product\_info}

class Price\_Product(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

product\_info=[]

spcl\_offer=ProductOffer.objects(productname=json\_data['prod\_name'])

details={}

if spcl\_offer.count()>0:

prod\_data=Sup\_Upload1.objects(upload\_name=spcl\_offer[0].productname)

for product in prod\_data:

print product.upload\_name

details['special\_disc']=spcl\_offer[0].discount

base=float(product.upload\_mrp)\*int(json\_data['qty'])

disc\_amt=base-(base\*float(product.upload\_discount))/100

spl\_disc=disc\_amt-(disc\_amt\*int(spcl\_offer[0].discount))/100

total=spl\_disc-(spl\_disc\*(int(json\_data['end\_user\_disc'])))/100

details['total']=total

saving=base-total

if saving==0:

details['saving\_percentage']=0

else:

details['saving\_p ercentage']=(saving\*100)/base

elif spcl\_offer.count()==0:

product\_data=Sup\_Upload1.objects(upload\_name=json\_data['prod\_name'])

for product in product\_data:

print product.upload\_name,'print product.upload\_namupload\_namee'

details['special\_disc']=0

spl\_dis=0

base=float(product.upload\_mrp)\*int(json\_data['qty'])

disc\_amt=base-(base\*float(product.upload\_discount))/100

spl\_disc=disc\_amt-(disc\_amt\*int(spl\_dis))/100

total=spl\_disc-(spl\_disc\*(int(json\_data['end\_user\_disc'])))/100

details['total']=total

saving=base-total

if saving==0:

details['saving\_percentage']=0

else:

details['saving\_percentage']=(saving\*100)/base

product\_info.append(details)

return {"status":"Success","product\_info":product\_info}

class VolumeDiscount(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

files = os.listdir("./static/ProductImages")

qty = request.args['qty']

intqty=int(qty)

prod\_name=request.args['upload\_name']

disc\_info=[]

discount=[]

for discoffer in Scheme.objects():

hg=int(discoffer.high)

lw=int(discoffer.low)

disc={}

disc['high']=discoffer.high

disc['low']=discoffer.low

disc['percentage']=discoffer.percentage

discount.append(disc)

if hg>=intqty and lw<=intqty:

details={}

details['percentage']=discoffer.percentage

disc\_info.append(details)

if qty=='0':

return {'status':'Success',"discount":discount}

product\_info=[]

spcl\_offer=ProductOffer.objects(productname=prod\_name)

details={}

if spcl\_offer.count()>0:

print "if"

details={}

prod\_data=Sup\_Upload1.objects(upload\_name=spcl\_offer[0].productname)

for product in prod\_data:

details['special\_disc']=format(float(spcl\_offer[0].discount),'.2f')

base=float(product.upload\_mrp)\*intqty

disc\_amt=base-base\*((float(product.upload\_discount)+float(spcl\_offer[0].discount))/100)

spl\_disc=0

for data in disc\_info:

spl\_disc=disc\_amt-disc\_amt\*(float(data['percentage'])/100)

details['vol\_Percentage']=disc\_info[0]['percentage']

details['total']=format(spl\_disc,'.2f')

details['qty']=qty

saving=base-spl\_disc

if saving==0:

details['saving\_percentage']=0

else:

details['saving\_percentage']=format((saving\*100)/base,'.2f')

elif spcl\_offer.count()==0:

print 'else'

product\_data=Sup\_Upload1.objects(upload\_name=prod\_name)

for product in product\_data:

base=float(product.upload\_mrp)\*intqty

details['special\_disc']='0'

disc\_amt=base-base\*((float(product.upload\_discount)+float(details['special\_disc']))/100)

spl\_disc=0

for data in disc\_info:

spl\_disc=disc\_amt-disc\_amt\*(float(data['percentage'])/100)

details['vol\_Percentage']=disc\_info[0]['percentage']

print details['vol\_Percentage']

details['total']=format(spl\_disc,'.2f')

details['qty']=qty

saving=base-spl\_disc

if saving==0:

details['saving\_percentage']=0

else:

details['saving\_percentage']=format((saving\*100)/base,'.2f')

product\_info.append(details)

upload\_data=Sup\_Upload1.objects.get(upload\_name=prod\_name)

upload\_data.salesqty=qty

upload\_data.save()

return {'status':'Success',"discount":discount,"product\_info":product\_info}

#----------------------------------------------RFQ GENERATION------------------

def synchronized(wrapped):

lock = threading.Lock()

@functools.wraps(wrapped)

def \_wrap(\*args, \*\*kwargs):

with lock:

return wrapped(\*args, \*\*kwargs)

return \_wrap

class RfqGenerateNo(Resource):

def appendzeros(self,package):

packagestr = ''

if package < 9:

packagestr = '000'+str(package)

elif package < 99:

packagestr = '00'+str(package)

elif package < 999:

packagestr = '0'+str(package)

else:

packagestr = str(package)

return packagestr

def getRfq(self,rfqNum):

presentRfqNum=''

package = 0

packageStr = ''

if rfqNum.startswith('RFQ'):

package = int(rfqNum[3:7])

package += 1

packageStr = self.appendzeros(package)

presentRfqNum ='RFQ'+ packageStr

else:

packageStr = self.appendzeros(1)

presentRfqNum = 'RFQ'+ packageStr

return presentRfqNum

@synchronized

def get(self):

lastRfq = LatestRfqNum.objects()

rfqReady=''

if lastRfq.count()==0:

rfqReady = self.getRfq('000000000000')

else:

rfqReady = self.getRfq(lastRfq[0].rfq\_num)

return rfqReady

@synchronized

def put(self, rfqCreated):

lastRfq = LatestRfqNum.objects()

if lastRfq.count()==0:

lastRfq = LatestRfqNum(rfq\_num=rfqCreated)

lastRfq.save()

else:

lastRfq = LatestRfqNum(id=lastRfq[0].id)

lastRfq.rfq\_num = rfqCreated

lastRfq.save()

return "put success"

def delete(self):

lastRfq = LatestRfqNum.objects()

lastRfq.delete()

return "success"

class NewRfq(Resource):

def get(self):

rfq\_num=RfqGenerateNo().get()

RfqGenerateNo().put(rfq\_num)

return {"status":"success","orderid":rfq\_num}

class Dealer\_Order\_Info(Resource):

def post(post):

json\_data=request.get\_json(force=True)

userid=json\_data['userid']

rfq\_no=RfqGenerateNo().get()

RfqGenerateNo().put(rfq\_no)

dealer\_order\_saving=DealerOrderData.objects(userid=userid,rfq\_num=rfq\_no)

if dealer\_order\_saving.count()==0:

deal\_name=Reg\_Dealer.objects.get(id=userid)

dealer\_info=DealerOrderData(userid=userid,rfq\_num=rfq\_no,rfq\_created\_by=deal\_name.dealer\_name,sales\_manager=deal\_name.sales\_contact\_person,sales\_manager\_mobile=deal\_name.sales\_contact\_mobile,po\_approved\_by='')

for data in json\_data['product\_info']:

productname=data['upload\_name']

product\_id=data['product\_id']

imageid=data['imageid']

modelno=data['upload\_modelno']

brand=data['brand']

qty=data['qty']

mrp=data['mrp']

discount=data['discount']

spcl\_disc=data['special\_discount']

volume\_disc=data['volume\_discount']

total=data['total']

savings=data['saving\_percentage']

save\_prod=Product\_Info(productname,product\_id,imageid,modelno,brand,qty,qty,mrp,discount,spcl\_disc,volume\_disc,total,savings)

dealer\_info.product\_info.append(save\_prod)

dealer\_info.save()

initiate\_order\_data=InitiateOrders.objects.get(user\_id=userid)

initiate\_order\_data.delete()

return {"status":"success","rfq\_no":rfq\_no,"userid":userid}

else:

return {"status":"rfq num already exists"}

class AddToCartSite(Resource):

def post(self):

json\_data = request.get\_json(force=True)

user\_id=json\_data['user\_id']

req\_randomnumber=json\_data['random\_no'] if 'random\_no' in json\_data else ''

orderitems\_list=json\_data['orderitem']

keyfunc = lambda x: x['productdescription']

groups = it.groupby(sorted(orderitems\_list, key=keyfunc), keyfunc)

final\_list= [{'productdescription':k, 'qty':sum(int(x['qty']) for x in g)} for k, g in groups]

if json\_data['user\_id']=='': #if userid empty

init\_order=InitiateOrdersSite.objects(user\_id=req\_randomnumber)

if init\_order.count() == 0:#if not random num

listofno = random.sample(set([1,2,3,4,5,6,7,8,9,'a','b','c','d','e','f','g']),16)

randomnumber="".join(str(numbers) for numbers in listofno)

print randomnumber,'randomnumber'

customer\_order=InitiateOrdersSite(user\_id=randomnumber)

for order\_items in final\_list:

productdescription=order\_items['productdescription']

qty=str(order\_items['qty'])

supp\_product=Sup\_Upload1.objects(upload\_name=productdescription).only('upload\_tax','upload\_id','prices','upload\_subcategory','upload\_brand','upload\_mrp','end\_user\_disc')

for ofrprice in supp\_product:

if float(ofrprice.upload\_mrp) != 0:

tax=ofrprice.upload\_tax

upload\_photo='011'+ofrprice.upload\_id+'.jpg'

image=upload\_photo

subcategory=ofrprice.upload\_subcategory

brand=ofrprice.upload\_brand

offer\_price = ofrprice.upload\_mrp

offer\_price\_gst=ofrprice.upload\_mrp

order\_details=OrderitemsSite(productdescription,qty,offer\_price,offer\_price\_gst,tax,image,subcategory,brand)

customer\_order.orderitemsite.append(order\_details)

customer\_order.save()

return {"status":"item added to cart","random\_number":randomnumber}

else:#if random num

order\_doc=InitiateOrdersSite.objects.get(user\_id=req\_randomnumber)

order\_doc.orderitemsite=[]

order\_doc.save()

for order\_items in final\_list:

productdescription=order\_items['productdescription']

qty=str(order\_items['qty'])

supp\_product=Sup\_Upload1.objects(upload\_name=productdescription).only('upload\_tax','upload\_id','prices','upload\_subcategory','upload\_brand','upload\_mrp','end\_user\_disc')

for ofrprice in supp\_product:

if float(ofrprice.upload\_mrp) != 0:

tax=ofrprice.upload\_tax

upload\_photo='011'+ofrprice.upload\_id+'.jpg'

image=upload\_photo

subcategory=ofrprice.upload\_subcategory

brand=ofrprice.upload\_brand

offer\_price = ofrprice.upload\_mrp

offer\_price\_gst=ofrprice.upload\_mrp

order\_details=OrderitemsSite(productdescription,qty,offer\_price,offer\_price\_gst,tax,image,subcategory,brand)

order\_doc.orderitemsite.append(order\_details)

order\_doc.save()

return {"status":"item added to cart","random\_number":req\_randomnumber}

if json\_data['user\_id'] != '': #if userid something

#print final\_list,"final\_list"

check\_cart=InitiateOrdersSite.objects(user\_id=user\_id)

orderitems\_list=json\_data['orderitem']

#var=[{"productdescription":"AAAA","qty":"1"},{"productdescription":"AAAA","qty":"1"}]

keyfunc = lambda x: x['productdescription']

groups = it.groupby(sorted(orderitems\_list, key=keyfunc), keyfunc)

final\_list= [{'productdescription':k, 'qty':sum(int(x['qty']) for x in g)} for k, g in groups]

if check\_cart.count() == 0:

customer\_order=InitiateOrdersSite(user\_id=user\_id)

for order\_items in final\_list:

productdescription=order\_items['productdescription']

qty=str(order\_items['qty'])

supp\_product=Sup\_Upload1.objects(upload\_name=productdescription).only('upload\_tax','upload\_id','prices','upload\_subcategory','upload\_brand','upload\_mrp','end\_user\_disc')

for ofrprice in supp\_product:

if float(ofrprice.upload\_mrp) != 0:

tax=ofrprice.upload\_tax

upload\_photo='011'+ofrprice.upload\_id+'.jpg'

image=upload\_photo

subcategory=ofrprice.upload\_subcategory

brand=ofrprice.upload\_brand

offer\_price = ofrprice.upload\_mrp

offer\_price\_gst=ofrprice.upload\_mrp

order\_details=OrderitemsSite(productdescription,qty,offer\_price,offer\_price\_gst,tax,image,subcategory,brand)

customer\_order.orderitemsite.append(order\_details)

customer\_order.save()

return {"status":"item added to cart"}

if check\_cart.count() > 0:

print "hello"

print final\_list,"final\_list final\_list"

order\_doc=InitiateOrdersSite.objects.get(user\_id=user\_id)

order\_doc.orderitemsite=[]

order\_doc.save()

print order\_doc.orderitemsite

for order\_items in final\_list:

productdescription=order\_items['productdescription']

print productdescription

qty=str(order\_items['qty'])

print qty

supp\_product=Sup\_Upload1.objects(upload\_name=productdescription).only('upload\_tax','upload\_id','prices','upload\_subcategory','upload\_brand','upload\_mrp','end\_user\_disc')

for ofrprice in supp\_product:

if float(ofrprice.upload\_mrp) != 0:

tax=ofrprice.upload\_tax

upload\_photo='011'+ofrprice.upload\_id+'.jpg'

image=upload\_photo

subcategory=ofrprice.upload\_subcategory

brand=ofrprice.upload\_brand

offer\_price = ofrprice.upload\_mrp

offer\_price\_gst=ofrprice.upload\_mrp

order\_details=OrderitemsSite(productdescription,qty,offer\_price,offer\_price\_gst,tax,image,subcategory,brand)

order\_doc.orderitemsite.append(order\_details)

order\_doc.save()

return {"status":"item added to cart"}

else:

return {"status":"invalid status"}

def put(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

product\_name=json\_data['product\_name']

update\_qty=json\_data['qty']

user\_id=json\_data['user\_id'] if 'user\_id' in json\_data else ''

randomnumber=json\_data['random\_no'] if 'random\_no' in json\_data else ''

if user\_id!='':

customer\_check=InitiateOrdersSite.objects.get(user\_id=user\_id)

for update in customer\_check.orderitemsite:

if update.productdescription==product\_name:

update.qty=str(update\_qty)

update.save()

customer\_check.save()

return {"status":"qty updated successfully"}

if randomnumber!='':

customer\_check=InitiateOrdersSite.objects.get(user\_id=randomnumber)

for update in customer\_check.orderitemsite:

if update.productdescription==product\_name:

update.qty=str(update\_qty)

update.save()

customer\_check.save()

return {"status":"qty updated successfully"}

else:

return {"status":"invalid user\_id"}

else:

return {"status":"invalid secret\_key"}

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args = request.args

files = os.listdir("./static/ProductImages")

userid=args['userid'] if 'userid' in args else ''

random\_no=args['random\_no'] if 'random\_no' in args else ''

#data1=InitiateOrders.objects.get(user\_id=random\_no)

if userid != '':

print 'am in first'

rand\_order\_detail=InitiateOrdersSite.objects(user\_id=userid)

if rand\_order\_detail.count()==0:

return {"status":"cart is empyt."}

total\_item\_list=[]

rand\_order\_details=InitiateOrdersSite.objects.get(user\_id=userid)

user\_id=rand\_order\_details.user\_id

for item in rand\_order\_details.orderitemsite:

data={}

data['upload\_name']=item.productdescription

data['qty']=item.qty

image=item.image

for prods in files:

if prods==image:

data['imageid']=prods

data['brand']=item.brand

data['mrp']=item.offer\_price

data['tax']=item.tax

data['subtotal']=float(data['qty'])\*float(data['mrp'])

data['tax\_amount']=(float(data['mrp'])\*(float(data['tax'])/100))\*float(data['qty'])

data['total\_inc\_gst']=item.offer\_price\_gst

json\_data=json.loads(json.dumps(data))

total\_item\_list.append(json\_data)

totalprice=sum(float(prod['total\_inc\_gst']) for prod in total\_item\_list)

total\_price=format(totalprice,'.2f')

return {"status":"Success","user\_id":userid,"total\_item\_list":total\_item\_list,"grand\_total":total\_price}

if random\_no =='' and userid == '':

return {"status":"cart is empyt."}

if random\_no !='':

total\_item\_list=[]

rand\_order\_details=InitiateOrdersSite.objects.get(user\_id=random\_no)

user\_id=rand\_order\_details.user\_id

for item in rand\_order\_details.orderitemsite:

data={}

data['upload\_name']=item.productdescription

data['qty']=item.qty

image=item.image

for prods in files:

if prods==image:

data['imageid']=prods

data['brand']=item.brand

data['tax']=item.tax

data['total']=item.offer\_price\_gst

data['mrp']=item.offer\_price

data['subtotal']=float(data['qty'])\*float(data['mrp'])

data['tax\_amount']=(float(data['mrp'])\*(float(data['tax'])/100))\*float(data['qty'])

json\_data=json.loads(json.dumps(data))

total\_item\_list.append(json\_data)

totalprice=sum(float(prod['total']) for prod in total\_item\_list)

total\_price=format(totalprice,'.2f')

return {"status":"Success","random\_no":random\_no,"user\_id":user\_id,"total\_item\_list":total\_item\_list,"total\_price":total\_price}

def delete(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args = request.args

json\_data = request.get\_json(force=True)

userid=args['userid'] if 'userid' in args else ''

rand\_num=args['random\_no'] if 'random\_no' in args else ''

print rand\_num,"rand\_num"

product=json\_data['product']

print product,"product"

if product == 'all' and userid!='' and rand\_num=='':

order\_del=InitiateOrdersSite.objects.get(user\_id=userid)

order\_del.orderitemsite=[]

order\_del.save()

if product != 'all' and userid!='' and rand\_num=='':

cart\_item = InitiateOrdersSite.objects.get(user\_id=userid)

a=[i for i,j in enumerate(cart\_item.orderitemsite) if j['productdescription'] == json\_data['product']][0]

print a

del cart\_item.orderitemsite[a]

cart\_item.save()

if len(cart\_item.orderitemsite)==0:

order\_del=InitiateOrdersSite.objects.get(user\_id=userid)

order\_del.delete()

if product == 'all' and rand\_num!='' and userid=='':

order\_del=InitiateOrdersSite.objects.get(user\_id=rand\_num)

order\_del.orderitemsite=[]

order\_del.save()

return {"status":"product deleted successfully"}

if product != 'all' and rand\_num!='' and userid=='':

cart\_item = InitiateOrdersSite.objects.get(user\_id=rand\_num)

#a=[i for i,j in enumerate(cart\_item.orderitemsite) if j['productdescription'] == json\_data['product']][0]

cart\_item.orderitemsite=[]

cart\_item.save()

return {"status":"product deleted successfully"}

return {"status":"product deleted successfully"}

else:

return {"status":"invalid secret key"}

'''

'''

from flask import Flask

from flask\_restful import Resource, Api

from smtservices import HomePageData,OffersProducts,NewArrivalsProducts,ProductDetails,Reviews,ProductlistService,Categories,SearchProduct,MatchProduct,Wishlist,YourWishlist,RemoveWishlist,\

Compare\_Prod\_Details\_post,InitiateOrder,Add\_Compare\_Product,Comp\_Prod\_Details\_get,Comp\_Prod\_Delete,Your\_Orders,Pending\_Orders,Invoices,Order\_Data,\

YourWishlistCartCount,AddToCartSite,Cancel\_Order,AltrProducts,SparProducts,CatSubBrand,CatSubcatBrandGet,VolumeDiscount,SubcategoryData,RfqGenerateNo,NewRfq,Dealer\_Order\_Info,Sub\_Brand,Price\_Product

from useradetails import Print\_Invoice\_Info,Inv\_Data,Proforma\_Inv\_Print,Chalan\_Details,AddressDefault,DealerVerifyOTP,DealerRegisterHistory,UserProfile,UsersDelete,InvGeneration,PoInformation,PODetails,DealerOrderConfirm,SplitData,UserLogin,UserRegistration,VerifyOTP,ResendOTP,LogOut,Dealer\_Registration,Admin\_Saving,\

Admin\_Login,DealerAcceptRejected,DealerRegisterData,Dealer\_Active,InternalusersLogout,AdminLogout,Dealer\_Login,Pending\_Rfq\_Get,POGenerateNo,User\_Saving,\

Users\_list,Pending\_PO\_Get,PoGeneration,RfqGet,DealerLogOut,InternalusersLogin,DealerOrderConfirm,DealerPODetails,InvChalan,Newchalan,DealerUserProfile,Estmate\_Avlqty\_Update,\

Rfq\_History,Proforma\_Inv\_Print,Company\_Details\_Saving,ChalanGeneration

from time import strftime

from xl import MasterUpload,AttributesUpload

from smtcheckout import Checkout,PayuData,PaymentStatus,ShopList,Pincode,DefaultAddress

newapp = Flask(\_\_name\_\_)

api = Api(newapp)

api.add\_resource(HomePageData, '/newhomepage')

api.add\_resource(OffersProducts, '/offersproducts')

api.add\_resource(NewArrivalsProducts, '/newarrivalsproducts')

api.add\_resource(ProductDetails, '/productdetails')

api.add\_resource(Reviews,'/reviews')

api.add\_resource(ProductlistService,'/newproductlist')

api.add\_resource(Categories,'/newcategories')

api.add\_resource(Wishlist,'/newwishlist')

api.add\_resource(YourWishlist,'/newyourwishlist')

api.add\_resource(RemoveWishlist,'/removewishlist')

api.add\_resource(SearchProduct,'/newsearchproduct')

api.add\_resource(MatchProduct,'/newmatchproduct')

api.add\_resource(Compare\_Prod\_Details\_post,'/compareproduct')

api.add\_resource(Add\_Compare\_Product,'/add\_compare\_product')

api.add\_resource(Comp\_Prod\_Details\_get,'/comp\_prod\_details')

api.add\_resource(Comp\_Prod\_Delete,'/comp\_prod\_delete')

api.add\_resource(InitiateOrder,'/dealer\_addtocart')

api.add\_resource(Your\_Orders, '/youroders')

api.add\_resource(Pending\_Orders, '/pendingorders')

api.add\_resource(Invoices, '/invoices')

api.add\_resource(Order\_Data, '/orderdata')

api.add\_resource(Cancel\_Order, '/cancelorder')

api.add\_resource(AltrProducts, '/alterproducts')

api.add\_resource(SparProducts, '/sparproducts')

api.add\_resource(CatSubBrand, '/catsubbrand')

api.add\_resource(SubcategoryData, '/subbrand')

api.add\_resource(CatSubcatBrandGet, '/catsubcatbrandget')

api.add\_resource(VolumeDiscount, '/volumediscount')

api.add\_resource(NewRfq, '/rfqnum')

api.add\_resource(UserRegistration, '/registration')

api.add\_resource(VerifyOTP, '/otpverify')

api.add\_resource(ResendOTP, '/resendotp')

api.add\_resource(UserLogin, '/login')

api.add\_resource(LogOut, '/logout')

api.add\_resource(Dealer\_Order\_Info, '/dealer\_order')

api.add\_resource(Dealer\_Registration, '/dealer\_register')

api.add\_resource(Admin\_Saving, '/admin')

api.add\_resource(Admin\_Login, '/admin\_login')

api.add\_resource(DealerAcceptRejected, '/dealer\_accept')

api.add\_resource(DealerRegisterData, '/dealer\_details')

api.add\_resource(DealerRegisterHistory, '/dealer\_history')

api.add\_resource(Dealer\_Active, '/dealer\_active')

api.add\_resource(Sub\_Brand, '/sub\_brand')

api.add\_resource(Dealer\_Login, '/dealer\_login\_logout')

api.add\_resource(Pending\_Rfq\_Get, '/rfq\_get')

api.add\_resource(Pending\_PO\_Get, '/po\_get')

api.add\_resource(RfqGet, '/rfq\_details')

api.add\_resource(PoGeneration, '/po\_generation')

api.add\_resource(DealerLogOut, '/dealer\_logout')

api.add\_resource(InternalusersLogin, '/sm\_login')

api.add\_resource(InternalusersLogout, '/sm\_logout')

api.add\_resource(AdminLogout, '/admin\_logout')

api.add\_resource(User\_Saving, '/users\_saving')

api.add\_resource(Users\_list, '/users\_details')

api.add\_resource(UsersDelete, '/users\_delete')

api.add\_resource(SplitData, '/split\_data')

api.add\_resource(DealerOrderConfirm, '/dealerorder\_confirm')

api.add\_resource(PODetails, '/po\_details')

api.add\_resource(DealerPODetails, '/dealer\_po\_details')

api.add\_resource(PoInformation, '/po\_information')

api.add\_resource(InvGeneration, '/inv\_generation')

api.add\_resource(InvChalan, '/invchalan\_info')

api.add\_resource(Newchalan, '/chalan')

api.add\_resource(DealerUserProfile, '/dealer\_profile')

api.add\_resource(Estmate\_Avlqty\_Update, '/estimate\_qty')

api.add\_resource(Rfq\_History, '/rfq\_history')

api.add\_resource(AddToCartSite, '/addtocart\_site')

api.add\_resource(Proforma\_Inv\_Print, '/pinv\_print')

api.add\_resource(Print\_Invoice\_Info, '/print\_invoice\_info')

api.add\_resource(Inv\_Data, '/invoice\_data')

api.add\_resource(Company\_Details\_Saving, '/company\_info')

api.add\_resource(ChalanGeneration, '/chalan\_info')

api.add\_resource(MasterUpload, '/masterdata\_upload')

api.add\_resource(AttributesUpload, '/attributes\_upload')

api.add\_resource(UserProfile, '/users\_profile')

api.add\_resource(DealerVerifyOTP, '/mobile\_update')

api.add\_resource(AddressDefault, '/setas\_default\_add')

api.add\_resource(Pincode,'/pincdechk')

api.add\_resource(Checkout,'/checkout')

api.add\_resource(ShopList,'/shoplist')

api.add\_resource(PayuData, '/payudata')

api.add\_resource(DefaultAddress,'/chkout\_defaultadd')

api.add\_resource(Chalan\_Details, '/chalan\_data')

@newapp.after\_request

def after\_request(response):

response.headers.add('Access-Control-Allow-Origin', '\*')

response.headers.add('Access-Control-Allow-Headers', 'Content-Type,Authorization,secret\_key')

response.headers.add('Access-Control-Allow-Methods', 'GET,PUT,POST,DELETE')

return response

if \_\_name\_\_ == '\_\_main\_\_':

newapp.run(host="192.168.20.107",port=1234)

from flask\_restful import Resource,Api

from flask\_httpauth import HTTPBasicAuth

from flask import jsonify, request

from werkzeug.security import check\_password\_hash,generate\_password\_hash

from models import Invoice\_Info,LatestInv,Product\_Info,CustomerDetails,CustomerSession,WishList,ProdInvShop,Reg\_Dealer,ConfirmInvoice,ShopUser,CustomerOrders,OrderItem,CustomerCoupon,Usertype\_Session,\

DuplicateCustomerDetails,InitiateOrders,Reg\_Dealer,Admin,Admin\_Session,Dealer\_Session,DealerOrderData,LatestPo,ProdinvWH,Usertype,Split\_Product\_Info,IvoiceMoreInfo,\

SplitInv,LatestChalanNo,Company\_Details,Sup\_Upload1,BillingAddress1,ShippingAddress1,UserSignup,Shopsetup,Sup\_Upload,CustBillingAddress,CustShippingAddress

from multiprocessing import Pool

import time

import json

import requests

import ast

from datetime import datetime,timedelta

from operator import itemgetter

import logging

from logging.handlers import RotatingFileHandler

from time import strftime

import traceback

import base64

import urllib

import onetimepass as otp

import threading

import functools

import random

import datetime

#from mongoengine.queryset.visitor import Q as mongo\_Q

from mongoengine.queryset.visitor import Q

import collections

from operator import itemgetter

import inflect

import math

secret\_key = '4r5t@W'

auth = HTTPBasicAuth()

USER\_DATA = {

"admin": "SuperSecretPwd"

}

def sendmail(email,subject,htmlbody):

value = requests.post(

"https://api.mailgun.net/v3/mg.shopmytools.com/messages",

auth=("api", "key-8bee70be27164355bdf6ad96ba54ca6d"),

data={"from": "SMT-Hub <info@mg.shopmytools.com>",

"to": email,

"subject": subject,

"html": htmlbody})

def sms(message,mobile):

url = "http://www.smscountry.com/smscwebservice\_bulk.aspx"

values = {'user' : 'powertex',

'passwd' : 'Powertexgst1',

'message':message,

'mobilenumber':mobile,

'mtype':'N',

'DR':'Y'

}

data=urllib.urlencode(values)

data = data.encode('utf-8')

f = urllib.urlopen(url, data)

print f.read().decode('utf-8')

log = logging.getLogger("\_\_name\_\_")

class Service(Api):

def handle\_error(self, e):

if not hasattr(e, 'data'):

e.data = e

ts = strftime('[%Y-%b-%d %H:%M:%S]')

tb = traceback.format\_exc()

log.error('%s %s %s %s %s 500 INTERNAL SERVER ERROR\n%s',

ts,

request.remote\_addr,

request.method,

request.scheme,

request.full\_path,

tb)

return super(Service, self).handle\_error(traceback.format\_exc(e))

@auth.verify\_password

def verify(username, password):

if '@' in username:

endUser = CustomerDetails.objects(email=username)

if endUser.count()==0:

return False

else:

endUser = ShopUser.objects(mobile=username)

if endUser.count()==0:

return False

dbPassword = str(endUser[0].password)

if check\_password\_hash(dbPassword,password):

return True

else:

return False

return USER\_DATA.get(username) == password

class UserRegistration(Resource):

def post(self):

json\_data = request.get\_json(force=True)

firstname = json\_data['firstname']

lastname = json\_data['lastname']

mobile = '91'+json\_data['mobile']

email = json\_data['email']

user\_type = json\_data['user\_type']

gst\_number = json\_data['gstnumber']

password=json\_data['password']

confirm\_password=json\_data['confirm\_password']

newsletter=json\_data['newsletter']

user\_type=json\_data['user\_type']

my\_secret = 'MFRGGZDFMZTWQ2LK'

otp\_is = otp.get\_totp(my\_secret)

print otp\_is,'otp'

if (CustomerDetails.objects(mobile=mobile)).count()>0 or (CustomerDetails.objects(email=email)).count()>0:

return {"status":"Mobile no / email already exists.."}

if DuplicateCustomerDetails.objects(mobile=mobile).count()>0 :

update\_details=DuplicateCustomerDetails.objects.get(mobile=mobile)

update\_details.firstname=firstname

update\_details.lastname=lastname

update\_details.mobile=mobile

update\_details.email=email

update\_details.gst\_number=gst\_number

update\_details.otp=str(otp\_is)

update\_details.status='inactive'

update\_details.newsletter=newsletter

update\_details.user\_type=user\_type

if password == confirm\_password :

update\_details.password=generate\_password\_hash(password)

msg='Your one time password is '+str(otp\_is)

mobile=mobile

print mobile

#pool.apply\_async(sms,[msg,mobile])

sms(msg,mobile)

print "hi"

update\_details.save()

return {"status":"Data saved successfully","otp":str(otp\_is)}

else:

return {"status":"password and confirm\_password does not matched"}

if password == confirm\_password :

print "regista"

user\_info=DuplicateCustomerDetails(firstname=firstname,lastname=lastname,mobile=mobile,email=email,password=generate\_password\_hash(password),gst\_number=gst\_number,otp=str(otp\_is),status='inactive',newsletter=newsletter,user\_type=user\_type)

user\_info.save()

msg='Your one time password is '+str(user\_info.otp)

mobile=str(user\_info.mobile)

#pool.apply\_async(sms,[msg,mobile])

sms(msg,mobile)

return {"status":"Data saved successfully","otp":str(user\_info.otp)}

else:

return {"status":"password and confirm\_password does not matched"}

class VerifyOTP(Resource):

def post(self):

json\_data = request.get\_json(force=True)

mobile='91'+json\_data['mobile']

otp=json\_data['otp']

ip\_address=json\_data['ip\_address']

check\_otp=DuplicateCustomerDetails.objects(mobile=mobile,otp=otp)

if check\_otp.count()>0:

customer\_details=DuplicateCustomerDetails.objects.get(id=check\_otp[0].id)

status=customer\_details.status='active'

firstname= customer\_details.firstname

lastname= customer\_details.lastname

mobile= customer\_details.mobile

email=customer\_details.email

password=customer\_details.password

gst\_number=customer\_details.gst\_number

otp=customer\_details.otp

newsletter=customer\_details.newsletter

user\_type=customer\_details.user\_type

if (CustomerDetails.objects(mobile=mobile)).count()>0 or (CustomerDetails.objects(email=email)).count()>0:

return {"status":"Mobile no / email already exists.."}

custmr\_save=CustomerDetails(firstname=firstname,lastname=lastname,mobile=mobile,email=email,password=password,gst\_number=gst\_number,otp=otp,status=status,user\_type=user\_type,ip\_address=ip\_address)

coupon = CustomerCoupon(coupon\_id='', createddate='',value='',coupon\_type='',maxvalue='')

custmr\_save.coupon\_info.append(coupon)

custmr\_save.save()

token=CustomerSession(mobile=mobile,email=email)

token.save()

if newsletter == 'checked':

fo = open("./static/templates/mail\_template.html", "r+")

htmlbody = fo.read()

fo.close()

urldata =custmr\_save.firstname.encode('ascii','ignore')+', thanks for registering to ShopMyTools.'

htmlbody = htmlbody.replace("$$urldata$$",urldata)

sendmail(sendMail,[custmr\_save.email,'Registration successful',htmlbody])

elif newsletter == 'unchecked':

print "newsletter unchecked"

del\_duplicate\_custmr=DuplicateCustomerDetails.objects(mobile=mobile)

del\_duplicate\_custmr=DuplicateCustomerDetails.objects(email=email)

del\_duplicate\_custmr.delete()

return {"status":"Data saved successfully","token":str(token.id),"user\_id":str(custmr\_save.id),"wishlist\_items":0,

"orders\_count":0,"invoice\_count":0,"pending\_order\_count":0,"reward\_points":0}

else :

return {"return":"invalid otp"}

class ResendOTP(Resource):

def get(self):

args = request.args

req\_mobile='91'+args['mobile']

get\_user=DuplicateCustomerDetails.objects.get(mobile=req\_mobile)

my\_secret = 'MFRGGZDFMZTWQ2LK'

otp\_is = otp.get\_totp(my\_secret)

get\_user.otp=str(otp\_is)

print get\_user.otp

get\_user.save()

msg='Your one time password is '+str(otp\_is)

sms(msg,req\_mobile)

return {"return":"otp sent to mobile"}

class UserLogin(Resource):

def post(self):

user=None

if request.headers.has\_key('Authorization'):

if request.headers['Authorization']:

json\_data = request.get\_json(force=True)

authentication\_value=request.headers['Authorization']

print authentication\_value

decode\_values=base64.b64decode(authentication\_value)

username=decode\_values.split(':')[0]

print authentication\_value

password=decode\_values.split(':')[1]

user\_info={}

if '@' in username:

user= CustomerDetails.objects(email=username,status='active')

if user.count()==0:

return {"status":"invalid user"}

user\_info['user\_mobile']=(user[0].mobile)[2:]

else:

user = CustomerDetails.objects(mobile='91'+username,status='active')

if user.count()==0:

return {"status":"invalid user"}

user\_info['user\_mobile']=(user[0].mobile)[2:]

if user=='':

return {"status":"Not a user"}

dbPassword = str(user[0].password)

pwd= check\_password\_hash(dbPassword,password)

if pwd:

if '@' in username:

token=CustomerSession(email=username)

token.save()

else:

token=CustomerSession(mobile=username)

token.save()

user\_info['email']=user[0].email

user\_info['firstname']=user[0].firstname

user\_info['lastname']=user[0].lastname

user\_info['user\_type']=user[0].user\_type

gst\_number=user[0].gst\_number

update\_type = CustomerDetails.objects.get(mobile=user[0].mobile,status='active')

update\_type.user\_type=json\_data['user\_type']

update\_type.ip\_address=json\_data['ip\_address'] if 'ip\_address' in json\_data else ''

update\_type.save()

final\_billing\_info={}

final\_shipping\_info={}

for value in user:

for billing\_addr in value.billing\_address:

billing\_info={"full\_name":billing\_addr.full\_name,"mobile":billing\_addr.mobile,"d\_no":billing\_addr.d\_no,"street":billing\_addr.street,"city":billing\_addr.city,"state":billing\_addr.state,"zipcode":billing\_addr.zipcode,"addunique\_nbr":billing\_addr.addunique\_nbr}

final\_billing\_info=billing\_info

for shipping\_addr in value.shipping\_address:

shipping\_info={"full\_name":shipping\_addr.full\_name,"mobile":shipping\_addr.mobile,"d\_no":shipping\_addr.d\_no,"street":shipping\_addr.street,"city":shipping\_addr.city,"state":shipping\_addr.state,"zipcode":shipping\_addr.zipcode,"addunique\_nbr":shipping\_addr.addunique\_nbr}

final\_shipping\_info=shipping\_info

return {"status":"Success","username":str(user[0].firstname),"user\_id":str(user[0].id),"token":str(token.id),"userinfo":user\_info,"billing\_address":final\_billing\_info,

"GSTnumber":gst\_number,"shipping\_address":final\_shipping\_info}

else:

return {"status":"Invalid Password"}

else:

return {"status":"enter valid credentials"}

else:

return {"status":"Authentication failed"}

class LogOut(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data = request.get\_json(force=True)

token = json\_data['token']

user = CustomerSession.objects.get(id=token)

user.delete()

return {"status":"success","message":"record deleted successfully"}

else:

return {"status":"invalid secret key"}

class SmtAcInfo(Resource):

@auth.login\_required

def put(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args = request.args

json\_data = request.get\_json(force=True)

req\_mobile='91'+json\_data['mobile']

user\_info = CustomerDetails.objects.get(id=args['id'])

if user\_info:

user\_info.firstname=json\_data['firstname']

user\_info.lastname =json\_data['lastname']

if req\_mobile :

check\_req\_mobile= CustomerDetails.objects(mobile=req\_mobile,id=args['id'])

if check\_req\_mobile.count() >0:

user\_info.mobile =req\_mobile

elif check\_req\_mobile.count() == 0:

check\_req\_mobile= CustomerDetails.objects(mobile=req\_mobile)

if check\_req\_mobile.count() >0:

return {"status":"mobile already exists"}

user\_info.mobile =req\_mobile

user\_info.save()

return {"status":"details updated successfully"}

else:

return {"msg":"invalid id"}

else:

return {"status":"invalid secret key"}

class ForgotPassword(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data = request.get\_json(force=True)

urldata=''

username=json\_data['username']

if '@' in username:

registeredUser = CustomerDetails.objects(email=json\_data['username'])

else:

registeredUser = CustomerDetails.objects(mobile='91'+json\_data['username'])

if registeredUser.count() > 0:

my\_secret = 'MFRGGZDFMZTWQ2LK'

otp\_is = otp.get\_totp(my\_secret)

user\_info=CustomerDetails.objects.get(mobile='91'+json\_data['username'])

user\_info.otp=str(otp\_is)

user\_info.save()

msg='Your one time password is '+str(otp\_is)

mobile='91'+username

pool.apply\_async(sms,[msg,mobile])

return {"status":"otp sent"}

if registeredUser.count()==0:

return {"status":"email/mobile does not exists"}

ts = int(round(time.time() \* 1000))

userid = registeredUser[0].id

fo = open("./static/smt-site-notifications/reset\_pwd.html", "r+")

htmlbody = fo.read()

fo.close()

restlint = '<a href="http://toolsomg.com/resetPassword.html#!/'+str(userid)+'",style="background-color: #068693; color: #ffffff; padding: 10px;">Reset Password</a>'

urldata =registeredUser[0].firstname.encode('ascii','ignore')+', Click on below reset password button to change password.<br>Link will be expired in 2hrs.<br>'+restlint

htmlbody = htmlbody.replace("$$urldata$$",urldata)

sendmail(sendMail,[registeredUser[0].email,'Reset password',htmlbody])

return {"status":"email sent","time\_stamp":str(ts)}

else:

return {"status":"invalid secret key"}

def get(self):

args = request.args

req\_mobile='91'+args['username']

req\_otp=args['otp']

user\_info = CustomerDetails.objects(mobile=req\_mobile,otp=req\_otp)

if user\_info.count() == 0:

return {"status":"invalid OTP"}

get\_otp=CustomerDetails.objects.get(mobile=req\_mobile,otp=req\_otp)

get\_otp.status='active'

get\_otp.save()

return {"status":"success","user\_id":str(get\_otp.id)}

class ResetPassword(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data = request.get\_json(force=True)

userid=args['user\_id']

new\_password=json\_data['new\_password']

confirm\_password=json\_data['confirm\_password']

time\_stamp=json\_data['time\_stamp']

user\_info = CustomerDetails.objects.get(id=userid)

if user\_info:

if time\_stamp != '':

pts = int(time\_stamp)

cts = int(round(time.time() \* 1000))

dts = cts-pts

hours=(dts/(1000\*60\*60))%24

if hours<2:

if new\_password == confirm\_password:

new\_pwd = generate\_password\_hash(new\_password)

user\_info.password=new\_pwd

user\_info.save()

return {"status":"password changed successfully"}

else:

return {"status":"new\_password and confirm\_password doesnot matched"}

else:

return {"status":"time out"}

elif time\_stamp == '':

if new\_password == confirm\_password:

new\_pwd = generate\_password\_hash(new\_password)

user\_info.password=new\_pwd

user\_info.save()

return {"status":"password changed successfully"}

else:

return {"status":"new\_password and confirm\_password doesnot matched"}

return {"status":"email does not exists"}

else:

return {"status":"invalid secret key"}

class Dealer\_Registration(Resource):

def post(self):

json\_data = request.get\_json(force=True)

user\_type=json\_data['user\_type']

dealer\_name = json\_data['dealer\_name']

dealer\_password=json\_data['dealer\_password']

dealer\_shopname=json\_data['dealer\_shopname']

reg\_shopname=json\_data['reg\_shopname']

dealer\_cin = json\_data['dealer\_cin']

dealer\_country= json\_data['dealer\_country']

dealer\_state= json\_data['dealer\_state']

dealer\_city = json\_data['dealer\_city']

dealer\_town= json\_data['dealer\_town']

dealer\_pin = json\_data['dealer\_pin']

dealer\_isd=json\_data['dealer\_isd']

dealer\_phone = json\_data['dealer\_phone']

dealer\_mail = json\_data['dealer\_mail']

dealer\_gstin=json\_data['dealer\_gstin']

dealer\_mobile=json\_data['dealer\_mobile']

dealer\_info=Reg\_Dealer.objects(dealer\_mail=dealer\_mail)

if dealer\_info.count()==0:

saving\_dealer\_info=Reg\_Dealer(user\_type=user\_type,dealer\_name=dealer\_name,dealer\_password=generate\_password\_hash(dealer\_password),dealer\_shopname=dealer\_shopname,reg\_shopname=reg\_shopname,dealer\_cin=dealer\_cin,dealer\_country=dealer\_country,dealer\_state=dealer\_state,dealer\_city=dealer\_city,dealer\_town=dealer\_town,dealer\_pin=dealer\_pin,dealer\_isd=dealer\_isd,dealer\_phone=dealer\_phone,dealer\_mail=dealer\_mail,dealer\_gstin=dealer\_gstin,dealer\_mobile=dealer\_mobile,turnover='',yr\_exp\_field='',spl\_remarks='',city\_category='',dealer\_category='',sales\_contact\_person='',sales\_contact\_mobile='',pan='',adhar\_num='',document\_verifiedby='',credit\_period\_days='',advance='').save()

mobile=dealer\_info[0].dealer\_isd+dealer\_mobile

msg='Thank you for your registration. Our Team will contact you soon for further process.'

sms(msg,mobile)

urldata ='Thank you for your registration. Our Team will contact you soon for further process.'

sendmail(sendmail,[dealer\_mail,'Registration successful',urldata])

return {"status":"dealer data saved successfully."}

else:

return {"status":"dealer already exists."}

class Admin\_Saving(Resource):

def post(self):

saving\_admin=Admin(admin\_mail='powertex.2017@gmail.com',user\_type='Admin',admin\_password=generate\_password\_hash('Powertex@123#')).save()

return {"status":"admin saved successfully."}

class Admin\_Login(Resource):

def post(self):

json\_data=request.get\_json(force=True)

admin\_mail=json\_data['mail']

password=json\_data['password']

if '@' in admin\_mail:

admin\_data=Admin.objects(admin\_mail=admin\_mail)

print admin\_data

if admin\_data.count()==0:

return {"status":"Login Fail"}

dbPassword=admin\_data[0].admin\_password

pwd=check\_password\_hash(dbPassword,password)

if pwd:

dealer\_list=[]

for dealers in Reg\_Dealer.objects():

print dealers

dealer\_data=json.loads(dealers.to\_json())

dealer\_list.append(dealer\_data)

admin\_session=Admin\_Session(admin\_mail=admin\_mail)

admin\_session.save()

return {"status":"success","user\_type":admin\_data[0].user\_type,"user\_name":admin\_data[0].user\_type,"token":str(admin\_session.id)}

else:

return {"status":" Password Fail"}

else:

admin\_data=Admin.objects(admin\_mail=admin\_mail,password=generate\_password\_hash(password))

if admin\_data.count()==0:

return {"status":"Fail"}

else:

for dealers in Reg\_Dealer.objects():

dealer\_data=json.loads(dealers.to\_json())

return {"sdeletetatus":"success","user\_type":admin\_data[0].user\_type,"token":str(admin\_session.id)}

class AdminLogout(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data = request.get\_json(force=True)

token = json\_data['token']

user = Admin\_Session.objects.get(id=token)

user.delete()

return {"status":"success","message":"record deleted successfully"}

else:

return {"status":"invalid secret key"}

class InternalusersLogin(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

user\_mail=json\_data['mail']

password=json\_data['password']

usertype=json\_data['usertype']

user\_data=Usertype.objects(user\_mail=user\_mail,usertype=usertype)

if user\_data.count()==0:

return {"status":"Login Fail"}

dbPassword=user\_data[0].user\_password

pwd=check\_password\_hash(dbPassword,password)

if pwd:

users\_list=[]

for users in Usertype.objects():

users\_data=json.loads(users.to\_json())

users\_list.append(users\_data)

users\_session=Usertype\_Session(user\_mail=user\_mail)

users\_session.save()

return {"status":"success","token":str(users\_session.id),"usertype":user\_data[0].usertype,"username":user\_data[0].username,"user\_id":str(user\_data[0].id)}

class InternalusersLogout(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data = request.get\_json(force=True)

token = json\_data['token']

user = Usertype\_Session.objects.get(id=token)

user.delete()

return {"status":"success","message":"Record deleted successfully"}

else:

return {"status":"invalid secret key"}

class DealerAcceptRejected(Resource):

def put(self):

json\_data = request.get\_json(force=True)

dealer\_mail=json\_data['dealer\_mail']

status=json\_data['status']

sales\_contact\_mobile=json\_data['sales\_contact\_mobile']

sales\_contact\_person=json\_data['sales\_contact\_person']

dealer\_category=json\_data['dealer\_category']

admin\_remarks=json\_data['admin\_remarks']

dealer\_info=Reg\_Dealer.objects(dealer\_mail=dealer\_mail,dealer\_status='Pending')

if dealer\_info.count()==0:

return {"status":"Dealer status already"}

else:

dealer\_data=Reg\_Dealer.objects.get(dealer\_mail=dealer\_mail,dealer\_status='Pending')

if status=='Accept':

dealer\_data.dealer\_status='Accept'

dealer\_data.sales\_contact\_mobile=sales\_contact\_mobile

dealer\_data.sales\_contact\_person=sales\_contact\_person

dealer\_data.sales\_contact\_person=dealer\_category

dealer\_data.sales\_contact\_person=admin\_remarks

dealer\_data.save()

return {"status":"Accepted"}

elif status=='Reject':

dealer\_data.dealer\_status='Reject'

dealer\_data.save()

return {"status":"Rejected"}

class DealerRegisterData(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

dealers\_list=[]

for dealer\_objects in Reg\_Dealer.objects(dealer\_status="Pending"):

dealers\_data=json.loads(dealer\_objects.to\_json())

dealers\_list.append(dealers\_data)

sm\_list=[]

for sm\_objects in Usertype.objects(usertype="salesManager"):

sm\_data=json.loads(sm\_objects.to\_json())

sm\_list.append(sm\_data)

categories=['A','B','C','D','E','F']

return {"status":"Success","Dealers\_info":dealers\_list,"Sm\_info":sm\_list,"categories":categories}

class Dealer\_Active(Resource):

def post(self):

json\_data=request.get\_json(force=True)

dealer\_mail=json\_data['dealer\_mail']

turnover=json\_data['turnover']

pan=json\_data['pan']

yr\_exp\_field=json\_data['yr\_exp\_field']

adhar\_num=json\_data['adhar\_num']

'''

document\_verifiedby=json\_data['document\_verifiedby']

advance=json\_data['advance']

reg\_shopname=json\_data['reg\_shopname']

if json\_data['credit\_period']!='':

advance='No'

credit\_period=json\_data['credit\_period']

if json\_data['advance']!='':

credit\_period='No'

advance=json\_data['advance']

'''

if '@' in dealer\_mail:

sales\_manager=Usertype.objects(usertype='salesManager')

dealer\_info=Reg\_Dealer.objects(dealer\_mail=dealer\_mail,dealer\_status='Accept')

print dealer\_info

if dealer\_info.count()>0:

for bill in json\_data['billing\_address']:

dno=bill['d\_no']

street=bill['street']

city=bill['city']

state=bill['state']

zipcode=bill['zip\_code']

for ship in json\_data['shipping\_address']:

dno=ship['d\_no']

street=ship['street']

city=ship['city']

state=ship['state']

zipcode=ship['zip\_code']

saving\_dealer\_info=Reg\_Dealer.objects.get(dealer\_mail=dealer\_mail)

bill\_addr=BillingAddress1(d\_no=str(dno),street=str(street),city=str(city),state=str(state),zipcode=str(zipcode))

ship\_addr=ShippingAddress1(d\_no=str(dno),street=str(street),city=str(city),state=str(state),zipcode=str(zipcode))

print ship\_addr,'ship'

saving\_dealer\_info.billing\_address.append(bill\_addr)

saving\_dealer\_info.shipping\_address.append(ship\_addr)

saving\_dealer\_info.save()

#saving\_dealer\_info.city\_category=city\_category

#saving\_dealer\_info.dealer\_category=dealer\_category

#saving\_dealer\_info.sales\_contact\_person=sales\_manager[0].username

#saving\_dealer\_info.sales\_contact\_mobile=sales\_manager[0].user\_mobile

saving\_dealer\_info.pan=pan

saving\_dealer\_info.year\_exp\_field=yr\_exp\_field

#saving\_dealer\_info.spl\_remarks=spl\_remarks

saving\_dealer\_info.turn\_over=turnover

saving\_dealer\_info.adhar\_num=adhar\_num

saving\_dealer\_info.document\_verifiedby=''

#saving\_dealer\_info.credit\_period\_days=credit\_period

#saving\_dealer\_info.advance=advance

#saving\_dealer\_info.reg\_shopname=reg\_shopname

saving\_dealer\_info.dealer\_status='Active'

saving\_dealer\_info.save()

mobile=saving\_dealer\_info.dealer\_isd+saving\_dealer\_info.dealer\_mobile

msg='Thank you for your registration. Our Team will contact you soon for further process.'

sms(msg,mobile)

urldata ='Thank you for your registration. Our Team will contact you soon for further process.'

sendmail(sendmail,[saving\_dealer\_info.dealer\_mail,'Registration successful',urldata])

return {"status":"Success","dealer\_status":"Dealer activated successfully."}

else:

print "else mail"

return {"status":"Dealer not exists."}

else:

sales\_manager=Usertype.objects(usertype='salesManager')

dealer\_info=Reg\_Dealer.objects(dealer\_mobile=dealer\_mail,dealer\_status='Accept')

if dealer\_info.count()>0:

for bill in json\_data['billing\_address']:

dno=bill['d\_no']

street=bill['street']

city=bill['city']

state=bill['state']

zipcode=bill['zip\_code']

for ship in json\_data['shipping\_address']:

dno=ship['d\_no']

street=ship['street']

city=ship['city']

state=ship['state']

zipcode=ship['zip\_code']

saving\_dealer\_info=Reg\_Dealer.objects.get(dealer\_mail=dealer\_mail)

bill\_addr=ShippingAddress1(d\_no=str(dno),street=str(street),city=str(city),state=str(state),zipcode=str(zipcode))

ship\_addr=BillingAddress1(d\_no=str(dno),street=str(street),city=str(city),state=str(state),zipcode=str(zipcode))

saving\_dealer\_info.billing\_address.append(bill\_addr)

saving\_dealer\_info.shipping\_address.append(ship\_addr)

#saving\_dealer\_info.city\_category=city\_category

#saving\_dealer\_info.dealer\_category=dealer\_category

#saving\_dealer\_info.sales\_contact\_person=sales\_manager[0].username

#saving\_dealer\_info.sales\_contact\_mobile=sales\_manager[0].user\_mobile

saving\_dealer\_info.pan=pan

saving\_dealer\_info.year\_exp\_field=yr\_exp\_field

#saving\_dealer\_info.spl\_remarks=spl\_remarks

saving\_dealer\_info.turn\_over=turnover

saving\_dealer\_info.adhar\_num=adhar\_num

saving\_dealer\_info.document\_verifiedby=''

#saving\_dealer\_info.credit\_period\_days=credit\_period

#saving\_dealer\_info.advance=advance

#saving\_dealer\_info.reg\_shopname=reg\_shopname

saving\_dealer\_info.dealer\_status='Active'

saving\_dealer\_info.save()

mobile=saving\_dealer\_info.dealer\_isd+saving\_dealer\_info.dealer\_mobile

msg='Thank you for your registration. Our Team will contact you soon for further process.'

sms(msg,mobile)

urldata ='Thank you for your registration. Our Team will contact you soon for further process.'

sendmail(sendmail,[saving\_dealer\_info.dealer\_mail,'Registration successful',urldata])

return {"status":"Success","dealer\_status":"Dealer activated successfully."}

else:

print "else mobile"

return {"status":"Dealer not exists."}

class Dealer\_Login(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

dealer\_mail=json\_data['mail']

password=json\_data['password']

user\_type=json\_data['user\_type']

if '@' in dealer\_mail:

dealer\_data=Reg\_Dealer.objects(dealer\_mail=dealer\_mail,user\_type=user\_type)#,password=generate\_password\_hash(password))

print dealer\_data

if dealer\_data.count()==0:

print 'hii'

return {"status":"Invalid User Credientials"}

user\_id=str(dealer\_data[0].id)

dealer\_status=dealer\_data[0].dealer\_status

dbPassword=dealer\_data[0].dealer\_password

pwd=check\_password\_hash(dbPassword,password)

dealer\_session=Dealer\_Session(dealer\_mail=dealer\_mail)

dealer\_session.save()

for dealers in Reg\_Dealer.objects(dealer\_mail=dealer\_mail,user\_type=user\_type):

dealer\_data=json.loads(dealers.to\_json())

return {"status":"success","dealer\_data":dealer\_data,"Dealer\_userid":user\_id,"Dealer\_status":dealer\_status,"token":str(dealer\_session.id)}

else:

dealer\_data=Reg\_Dealer.objects(dealer\_mobile=dealer\_mail,password=generate\_password\_hash(password))

user\_id=str(dealer\_data[0].id)

if dealer\_data.count()==0:

return {"status":"Invalid User Credientials"}

else:

if dealer\_data[0].dealer\_status=='Accept':

dealer\_status='Accepted'

else:

dealer\_status='Pending'

dbPassword=dealer\_data[0].dealer\_password

pwd=Reg\_Dealercheck\_password\_hash(dbPassword,password)

dealer\_session=Dealer\_Session(dealer\_mail=dealer\_mail)

dealer\_session.save()

for dealers in Reg\_Dealer.objects(dealer\_mobile=dealer\_mail,user\_type=user\_type):

dealer\_data=json.loads(dealers.to\_json())

return {"status":"success","dealer\_data":dealer\_data,"Dealer\_userid":user\_id,"Dealer\_status":dealer\_status,"token":str(dealer\_session.id)}

class DealerLogOut(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data = request.get\_json(force=True)

token = json\_data['token']

user = Dealer\_Session.objects.get(id=token)

user.delete()

return {"status":"success","message":"record deleted successfully"}

else:

return {"status":"invalid secret key"}

class DealerUserProfile(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

userid=request.args['userid']

dealer\_data=Reg\_Dealer.objects.get(id=userid).only('dealer\_mobile','dealer\_name','dealer\_mail','billing\_address','shipping\_address')

data=json.loads(dealer\_data.to\_json())

return{"status":"success","Dealer\_profile":data}

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

bill\_address=json\_data['billing\_address']

ship\_address=json\_data['shipping\_address']

userid=json\_data['userid']

if bill\_address !='' and ship\_address == '':

dealer\_bill\_data=Reg\_Dealer.objects(id=userid,billing\_address\_\_in=[bill\_address])

if dealer\_bill\_data.count()>0:

dealer\_bill\_info=Reg\_Dealer.objects.get(id=userid,billing\_address\_\_in=[bill\_address])

for bill in dealer\_bill\_info.billing\_address:

bill=[bill\_address]

dealer\_bill\_info.save()

else:

dealer\_info=Reg\_Dealer.objects.get(id=userid)

dealer\_info.billing\_address.append(bill\_address)

dealer\_info.save()

if ship\_address != '' and bill\_address =='':

dealer\_ship\_data=Reg\_Dealer.objects(id=userid,shipping\_address\_\_in=[ship\_address])

if dealer\_ship\_data.count()>0:

dealer\_info=Reg\_Dealer.objects.get(id=userid,shipping\_address\_\_in=[ship\_address])

for ship in dealer\_info.shipping\_address:

ship=[ship\_address]

dealer\_info.save()

else:

dealer\_info=Reg\_Dealer.objects.get(id=userid)

dealer\_info.shipping\_address.append(ship\_address)

dealer\_info.save()

if ship\_address != '' and bill\_address !='':

dealer\_bill\_data=Reg\_Dealer.objects(id=userid,billing\_address\_\_in=[bill\_address])

if dealer\_bill\_data.count()>0:

dealer\_bill\_info=Reg\_Dealer.objects.get(id=userid,billing\_address\_\_in=[bill\_address])

for bill in dealer\_bill\_info.billing\_address:

bill=[bill\_address]

dealer\_bill\_info.save()

else:

dealer\_info=Reg\_Dealer.objects.get(id=userid)

dealer\_info.billing\_address.append(bill\_address)

dealer\_info.save()

dealer\_ship\_data=Reg\_Dealer.objects(id=userid,shipping\_address\_\_in=[ship\_address])

if dealer\_ship\_data.count()>0:

dealer\_info=Reg\_Dealer.objects.get(id=userid,shipping\_address\_\_in=[ship\_address])

for ship in dealer\_info.shipping\_address:

ship=[ship\_address]

dealer\_info.save()

else:

dealer\_info=Reg\_Dealer.objects.get(id=userid)

dealer\_info.shipping\_address.append(ship\_address)

dealer\_info.save()

return{"status":"success"}

def delete(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

bill\_address=json\_data['billing\_address']

ship\_address=json\_data['shipping\_address']

userid=json\_data['userid']

if bill\_address!='':

dealer\_bill\_data=Reg\_Dealer.objects(id=userid,billing\_address\_\_in=[bill\_address])

if dealer\_bill\_data.count()>0:

dealer\_bill\_info=Reg\_Dealer.objects.get(id=userid,billing\_address\_\_in=[bill\_address])

for bill in dealer\_bill\_info.billing\_address:

if bill==bill\_address:

dealer\_bill\_info.billing\_address.remove(bill)

dealer\_bill\_info.save()

if ship\_address!='':

dealer\_ship\_data=Reg\_Dealer.objects(id=userid,shipping\_address\_\_in=[ship\_address])

if dealer\_ship\_data.count()>0:

dealer\_info=Reg\_Dealer.objects.get(id=userid,shipping\_address\_\_in=[ship\_address])

for ship in dealer\_info.shipping\_address:

if ship==ship\_address:

dealer\_info.shipping\_address.remove(ship)

dealer\_info.save()

if ship\_address != '' and bill\_address !='':

dealer\_bill\_data=Reg\_Dealer.objects(id=userid,billing\_address\_\_in=[bill\_address])

if dealer\_bill\_data.count()>0:

dealer\_bill\_info=Reg\_Dealer.objects.get(id=userid,billing\_address\_\_in=[bill\_address])

for bill in dealer\_bill\_info.billing\_address:

if bill==bill\_address:

dealer\_bill\_info.billing\_address.remove(bill)

dealer\_bill\_info.save()

dealer\_ship\_data=Reg\_Dealer.objects(id=userid,shipping\_address\_\_in=[ship\_address])

if dealer\_ship\_data.count()>0:

dealer\_info=Reg\_Dealer.objects.get(id=userid,shipping\_address\_\_in=[ship\_address])

for ship in dealer\_info.shipping\_address:

if ship==ship\_address:

dealer\_info.shipping\_address.remove(ship)

dealer\_info.save()

return{"status":"success"}

class Pending\_Rfq\_Get(Resource):

def get(self):

usertype=request.args['usertype']

userid=request.args['userid']

if usertype=='salesManager':

rfq\_list=[]

sales\_name = Usertype.objects.get(usertype='salesManager',id=userid)

sales\_check=DealerOrderData.objects(sales\_manager\_mobile=sales\_name.user\_mobile,status="Pending")

if sales\_check.count()==0:

return {"status":"no records found"}

for rfq\_objects in DealerOrderData.objects(sales\_manager\_mobile=sales\_name.user\_mobile,status="Pending"):

dealer\_info=Reg\_Dealer.objects.get(id=rfq\_objects.userid)

details2={}

details2["dealer\_name"]=dealer\_info.dealer\_name

details2["dealer\_city"]=dealer\_info.dealer\_city

details2["dealer\_mail"]=dealer\_info.dealer\_mail

details2["dealer\_mobile"]=dealer\_info.dealer\_mobile

details2['rfq\_generate\_date']=(rfq\_objects.rfq\_generate\_date).strftime("%d-%m-%Y")

details2["rfq\_num"]=rfq\_objects.rfq\_num

more\_info=[]

for data in rfq\_objects.product\_info:

details={}

details["productid"]=data.productid

details["productname"]=data.productname

details["modelno"]=data.modelno

details["brand"]=data.brand

details["qty"]=data.qty

details["mrp"]=data.mrp

details["total"]=data.total

details["discount"]=data.discount

details["rfq\_num"]=rfq\_objects.rfq\_num

details["volume\_disc"]=data.volume\_disc

details["savings"]=data.savings

if data.status=='Pending' or len(data.split\_products)==0:

details["product\_status"]='Pending for SM'

split\_list=[]

for k in data.split\_products:

details["spcl\_disc"]=data.spcl\_disc

details["accept\_productname"]=k.accept\_productname

details["balance\_qty"]=k.balance\_qty

details["split\_status"]=k.split\_status

details["product\_status"]=k.product\_status

details["dealer\_status"]=k.dealer\_status

details["unique\_nbr"]=k.unique\_nbr

details["rfq\_num"]=rfq\_objects.rfq\_num

details["split\_delvry\_date"]=k.split\_delvry\_date

if data.productname==k.accept\_productname and k.product\_status=='Split':

info={"remarks\_dealer":k.remarks\_dealer,"remarks\_internal":k.remarks\_internal,"product\_status":k.product\_status,"productname":k.accept\_productname,"rfq\_num":rfq\_objects.rfq\_num,"split\_delvry\_date":k.split\_delvry\_date,"unique\_nbr":k.unique\_nbr,"accept\_prod\_qty":k.accept\_prod\_qty}

split\_list.append(info)

details["split\_info"]=split\_list

more\_info.append(details)

details2["rfq\_list"]=more\_info

rfq\_list.append(details2)

newlist = sorted(rfq\_list, key=lambda k: k['rfq\_num'])

return {"status":"Success","more\_info":newlist}

if usertype=='Dealer':

deal\_check=DealerOrderData.objects(userid=userid,status="Pending")

if deal\_check.count()==0:

return {"status":"no records found"}

rfq\_list=[]

for rfqnum in DealerOrderData.objects(userid=userid,status="Pending"):

sales\_check=Usertype.objects(usertype='salesManager',user\_mobile=rfqnum.sales\_manager\_mobile)

print sales\_check

if sales\_check.count()==0:

return {"status":"sales manager not assgined to dealer"}

sales\_name=Usertype.objects.get(usertype='salesManager',user\_mobile=rfqnum.sales\_manager\_mobile)

details2={}

details2['sales\_manager']=sales\_name.username

details2['sales\_manager\_mobile']=sales\_name.user\_mobile

details2['rfq\_num']=rfqnum.rfq\_num

details2['sales\_manager\_mobile']=rfqnum.sales\_manager\_mobile

details2['rfq\_generate\_date']=(rfqnum.rfq\_generate\_date).strftime("%d-%m-%Y")

details2['remarks']=rfqnum.remarks

more\_info=[]

for data in rfqnum.product\_info:

details={}

details["productid"]=data.productid

details["productname"]=data.productname

details["modelno"]=data.modelno

details["brand"]=data.brand

details["qty"]=data.qty

details["mrp"]=data.mrp

details["total"]=data.total

details["discount"]=data.discount

details["rfq\_num"]=rfqnum.rfq\_num

details["volume\_disc"]=data.volume\_disc

details["savings"]=data.savings

if data.status=='Pending' or len(data.split\_products)==0:

details["product\_status"]='Pending for SM'

split\_list=[]

for k in data.split\_products:

details["spcl\_disc"]=data.spcl\_disc

details["accept\_productname"]=k.accept\_productname

details["balance\_qty"]=k.balance\_qty

details["split\_status"]=k.split\_status

details["product\_status"]=k.product\_status

details["dealer\_status"]=k.dealer\_status

details["unique\_nbr"]=k.unique\_nbr

details["rfq\_num"]=rfqnum.rfq\_num

details["split\_delvry\_date"]=k.split\_delvry\_date

if data.productname==k.accept\_productname and k.product\_status=='Split':

info={"remarks\_dealer":k.remarks\_dealer,"remarks\_internal":k.remarks\_internal,"product\_status":k.product\_status,"productname":k.accept\_productname,"rfq\_num":rfqnum.rfq\_num,"split\_delvry\_date":k.split\_delvry\_date,"unique\_nbr":k.unique\_nbr,"accept\_prod\_qty":k.accept\_prod\_qty}

split\_list.append(info)

details["split\_info"]=split\_list

more\_info.append(details)

action=''

action\_list=[]

for f in more\_info:

if 'product\_status' in f:

action\_list.append(f['product\_status'])

if 'Pending for SM' in action\_list:

action='Disable'

if 'Pending for SM' not in action\_list:

action='Enable'

details2["action"]=action

details2["rfq\_list"]=more\_info

rfq\_list.append(details2)

newlist = sorted(rfq\_list, key=lambda k: k['rfq\_num'])

#print newlist

return {"status":"Success","more\_info":newlist}

if usertype=='accountant':

deal\_check=DealerOrderData.objects()

if deal\_check.count()==0:

return {"status":"no records found"}

rfq\_list=[]

for rfqnum in DealerOrderData.objects():

sales\_name=Usertype.objects.get(usertype='salesManager',user\_mobile=rfqnum.sales\_manager\_mobile)

acnt\_name=Usertype.objects.get(id=userid)

dealer\_info=Reg\_Dealer.objects.get(id=rfqnum.userid)

details2={}

details2['sales\_manager']=sales\_name.username

details2['dealer\_name']=dealer\_info.dealer\_name

details2['dealer\_mail']=dealer\_info.dealer\_mail

details2['dealer\_mobile']=dealer\_info.dealer\_mobile

details2['dealer\_city']=dealer\_info.dealer\_city

details2['acnt\_mail']=acnt\_name.user\_mail

details2['acnt\_mobile']=acnt\_name.user\_mobile

details2['acnt\_name']=acnt\_name.username

details2['sales\_manager\_mobile']=sales\_name.user\_mobile

details2['rfq\_num']=rfqnum.rfq\_num

details2['status']=rfqnum.status

details2['po\_num']=rfqnum.po\_num

details2['status']=rfqnum.status

details2['po\_gen\_date']=rfqnum.po\_gen\_date

details2['sales\_manager\_mobile']=rfqnum.sales\_manager\_mobile

details2['rfq\_generate\_date']=(rfqnum.rfq\_generate\_date).strftime("%d-%m-%Y")

details2['remarks']=rfqnum.remarks

more\_info=[]

for data in rfqnum.product\_info:

details={}

details["productid"]=data.productid

details["productname"]=data.productname

details["modelno"]=data.modelno

details["brand"]=data.brand

details["qty"]=data.qty

details["mrp"]=data.mrp

details["total"]=data.total

details["discount"]=data.discount

details["rfq\_num"]=rfqnum.rfq\_num

details["volume\_disc"]=data.volume\_disc

details["savings"]=data.savings

if data.status=='Pending' or len(data.split\_products)==0:

details["product\_status"]='Pending for SM'

split\_list=[]

for k in data.split\_products:

details["spcl\_disc"]=data.spcl\_disc

details["accept\_productname"]=k.accept\_productname

details["balance\_qty"]=k.balance\_qty

details["split\_status"]=k.split\_status

details["product\_status"]=k.product\_status

details["dealer\_status"]=k.dealer\_status

details["unique\_nbr"]=k.unique\_nbr

details["rfq\_num"]=rfqnum.rfq\_num

details["split\_delvry\_date"]=k.split\_delvry\_date

if data.productname==k.accept\_productname and k.product\_status=='Split':

info={"remarks\_dealer":k.remarks\_dealer,"remarks\_internal":k.remarks\_internal,"product\_status":k.product\_status,"productname":k.accept\_productname,"rfq\_num":rfqnum.rfq\_num,"split\_delvry\_date":k.split\_delvry\_date,"unique\_nbr":k.unique\_nbr,"accept\_prod\_qty":k.accept\_prod\_qty}

split\_list.append(info)

details["split\_info"]=split\_list

more\_info.append(details)

details2["rfq\_list"]=more\_info

rfq\_list.append(details2)

newlist = sorted(rfq\_list, key=lambda k: k['rfq\_num'])

return {"status":"Success","more\_info":newlist}

class DealerOrderConfirm(Resource):#PO-Generations

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data = request.get\_json(force=True)

data= json\_data['data']

rfq\_num= json\_data['rfq\_num']

po\_num=POGenerateNo().get()

user\_data=DealerOrderData.objects(rfq\_num=rfq\_num)

if user\_data.count()==0:

return {"status":"rfq doesnot exists"}

for prod in user\_data:

for info in prod.product\_info:

for split in info.split\_products:

for num in data:

if split.accept\_productname==num['product\_name'] and split.unique\_nbr==num['unique\_nbr']:

split.dealer\_status=num['dealer\_status']

split.save()

if split.accept\_productname==num['product\_name'] and num['unique\_nbr']=='':

split.dealer\_status=num['dealer\_status']

split.save()

date=datetime.date.today().strftime("%d-%m-%Y")

prod.po\_num=po\_num

prod.status='PO Generated'

prod.po\_gen\_date=str(date)

num=DealerOrderData.objects.get(rfq\_num=rfq\_num)

sms\_num=Reg\_Dealer.objects.get(id=num.userid)

msg='Purchase Order No: '+po\_num+' created.'

mobile=str(sms\_num.dealer\_mobile)

sms(msg,mobile)

urldata ='Your PO No. '+po\_num+' has been generated. Sales Manager will accept your PO no. and we will inform you soon.'

sendmail(sendmail,[sms\_num.dealer\_mail,'PO Generation Successful'],urldata)

prod.save()

POGenerateNo().put(po\_num)

return{"status":"Success","Po\_number":po\_num,"rfq\_number":rfq\_num}

class PODetails(Resource):#SALESMANAGER

def get(self):

#if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

userid=request.args['userid']

usertype=request.args['usertype']

if usertype=='salesManager':

rfq\_list=[]

print 'hi'

sales\_name = Usertype.objects.get(usertype='salesManager',id=userid)

deal\_check=DealerOrderData.objects(sales\_manager\_mobile=sales\_name.user\_mobile,status="PO Generated")

if deal\_check.count()==0:

return {"status":"no records found"}

for rfq\_objects in DealerOrderData.objects(sales\_manager\_mobile=sales\_name.user\_mobile,status="PO Generated").filter(product\_info\_\_split\_products\_\_match={'dealer\_status':'Confirm'}):

dealer\_info=Reg\_Dealer.objects.get(id=rfq\_objects.userid)

details1={}

details1["dealer\_name"]=dealer\_info.dealer\_name

details1["dealer\_city"]=dealer\_info.dealer\_city

details1["dealer\_mail"]=dealer\_info.dealer\_mail

details1["dealer\_mobile"]=dealer\_info.dealer\_mobile

details1['rfq\_num']=rfq\_objects.rfq\_num

details1['status']=rfq\_objects.status

details1['po\_num']=rfq\_objects.po\_num

details1['total']=rfq\_objects.product\_info[0].total

details1['rfq\_generate\_date']=(rfq\_objects.rfq\_generate\_date).strftime("%d-%m-%Y")

details1['po\_gen\_date']=rfq\_objects.po\_gen\_date

more\_info=[]

for data in rfq\_objects.product\_info:

details={}

if data.status=='Pending' or len(data.split\_products)==0:

details["product\_status"]='Pending for PO'

split\_list=[]

for k in data.split\_products:

if data.productname==k.accept\_productname and k.split\_status=='Accept':

details["productid"]=data.productid

details["productname"]=data.productname

details["modelno"]=data.modelno

details["brand"]=data.brand

details["qty"]=data.qty

details["mrp"]=data.mrp

details["total"]=data.total

details["discount"]=data.discount

details["rfq\_num"]=rfq\_objects.rfq\_num

details["volume\_disc"]=data.volume\_disc

details["savings"]=data.savings

details["spcl\_disc"]=data.spcl\_disc

details["accept\_productname"]=k.accept\_productname

details["balance\_qty"]=k.balance\_qty

details["split\_status"]=k.split\_status

details["product\_status"]=k.product\_status

details["dealer\_status"]=k.dealer\_status

details["unique\_nbr"]=k.unique\_nbr

details["rfq\_num"]=rfq\_objects.rfq\_num

details["split\_delvry\_date"]=k.split\_delvry\_date

if data.productname==k.accept\_productname and k.product\_status=='Accept' or k.product\_status=='Split':

info={"remarks\_dealer":k.remarks\_dealer,"product\_status":k.product\_status,"remarks\_internal":k.remarks\_internal,"productname":k.accept\_productname,"rfq\_num":rfq\_objects.rfq\_num,"split\_delvry\_date":k.split\_delvry\_date,"unique\_nbr":k.unique\_nbr,"accept\_prod\_qty":k.accept\_prod\_qty}

split\_list.append(info)

details["split\_info"]=split\_list

more\_info.append(details)

details1["rfq\_list"]=more\_info

rfq\_list.append(details1)

newlist = sorted(rfq\_list, key=lambda k: k['rfq\_num'])

return {"status":"Success","more\_info":newlist}

if usertype=='Dealer':

deal\_check=DealerOrderData.objects(userid=userid,status="PO Generated")

if deal\_check.count()==0:

return {"status":"no records found"}

rfq\_list=[]

for rfqnum in DealerOrderData.objects(userid=userid,status="PO Generated"):

sales\_name=Usertype.objects.get(usertype='salesManager',user\_mobile=rfqnum.sales\_manager\_mobile)

details2={}

details2['sales\_manager']=sales\_name.username

details2['sales\_manager\_mobile']=sales\_name.user\_mobile

details2['rfq\_num']=rfqnum.rfq\_num

details2['po\_num']=rfqnum.po\_num

details2['status']=rfqnum.status

details2['po\_gen\_date']=rfqnum.po\_gen\_date

details2['sales\_manager\_mobile']=rfqnum.sales\_manager\_mobile

details2['rfq\_generate\_date']=(rfqnum.rfq\_generate\_date).strftime("%d-%m-%Y")

details2['remarks']=rfqnum.remarks

more\_info=[]

for data in rfqnum.product\_info:

details={}

if data.status=='Pending' or len(data.split\_products)==0:

details["product\_status"]='Pending for PO'

split\_list=[]

for k in data.split\_products:

if data.productname==k.accept\_productname and k.split\_status=='Accept':

details["productid"]=data.productid

details["productname"]=data.productname

details["modelno"]=data.modelno

details["brand"]=data.brand

details["qty"]=data.qty

details["mrp"]=data.mrp

details["total"]=data.total

details["discount"]=data.discount

details["rfq\_num"]=rfqnum.rfq\_num

details["volume\_disc"]=data.volume\_disc

details["savings"]=data.savings

details["spcl\_disc"]=data.spcl\_disc

details["accept\_productname"]=k.accept\_productname

details["balance\_qty"]=k.balance\_qty

details["split\_status"]=k.split\_status

details["product\_status"]=k.product\_status

details["dealer\_status"]=k.dealer\_status

details["unique\_nbr"]=k.unique\_nbr

details["rfq\_num"]=rfqnum.rfq\_num

details["split\_delvry\_date"]=k.split\_delvry\_date

if data.productname==k.accept\_productname and k.product\_status=='Split' or k.product\_status=='Accept':

info={"remarks\_dealer":k.remarks\_dealer,"remarks\_internal":k.remarks\_internal,"product\_status":k.product\_status,"productname":k.accept\_productname,"rfq\_num":rfqnum.rfq\_num,"split\_delvry\_date":k.split\_delvry\_date,"unique\_nbr":k.unique\_nbr,"accept\_prod\_qty":k.accept\_prod\_qty}

split\_list.append(info)

details["split\_info"]=split\_list

more\_info.append(details)

details2["rfq\_list"]=more\_info

rfq\_list.append(details2)

newlist = sorted(rfq\_list, key=lambda k: k['rfq\_num'])

return {"status":"Success","more\_info":newlist}

if usertype=='accountant' or usertype=='whManager':

deal\_check=DealerOrderData.objects(status='Approved')

if deal\_check.count()==0:

return {"status":"no records found"}

rfq\_list=[]

for rfqnum in DealerOrderData.objects(status='Approved'):

sales\_name=Usertype.objects.get(usertype='salesManager',user\_mobile=rfqnum.sales\_manager\_mobile)

acnt\_name=Usertype.objects.get(id=userid)

dealer\_info=Reg\_Dealer.objects.get(id=rfqnum.userid)

details2={}

details2['sales\_manager']=sales\_name.username

details2['dealer\_name']=dealer\_info.dealer\_name

details2['dealer\_mail']=dealer\_info.dealer\_mail

details2['dealer\_mobile']=dealer\_info.dealer\_mobile

details2['dealer\_city']=dealer\_info.dealer\_city

details2['acnt\_mail']=acnt\_name.user\_mail

details2['acnt\_mobile']=acnt\_name.user\_mobile

details2['acnt\_name']=acnt\_name.username

details2['sales\_manager\_mobile']=sales\_name.user\_mobile

details2['rfq\_num']=rfqnum.rfq\_num

details2['status']=rfqnum.status

details2['po\_num']=rfqnum.po\_num

details2['status']=rfqnum.status

details2['po\_gen\_date']=rfqnum.po\_gen\_date

details2['sales\_manager\_mobile']=rfqnum.sales\_manager\_mobile

details2['rfq\_generate\_date']=(rfqnum.rfq\_generate\_date).strftime("%d-%m-%Y")

details2['remarks']=rfqnum.remarks

more\_info=[]

for data in rfqnum.product\_info:

details={}

if data.status=='Pending' or len(data.split\_products)==0:

details["product\_status"]='Pending for PO'

split\_list=[]

for k in data.split\_products:

if data.productname==k.accept\_productname and k.split\_status=='Accept':

details["productid"]=data.productid

details["productname"]=data.productname

details["modelno"]=data.modelno

details["brand"]=data.brand

details["qty"]=data.qty

details["mrp"]=data.mrp

details["total"]=data.total

details["discount"]=data.discount

details["rfq\_num"]=rfqnum.rfq\_num

details["volume\_disc"]=data.volume\_disc

details["savings"]=data.savings

details["spcl\_disc"]=data.spcl\_disc

details["accept\_productname"]=k.accept\_productname

details["balance\_qty"]=k.balance\_qty

details["split\_status"]=k.split\_status

details["product\_status"]=k.product\_status

details["dealer\_status"]=k.dealer\_status

details["unique\_nbr"]=k.unique\_nbr

details["rfq\_num"]=rfqnum.rfq\_num

details["split\_delvry\_date"]=k.split\_delvry\_date

if data.productname==k.accept\_productname and k.product\_status=='Split' or k.product\_status=='Accept':

info={"remarks\_dealer":k.remarks\_dealer,"remarks\_internal":k.remarks\_internal,"product\_status":k.product\_status,"productname":k.accept\_productname,"rfq\_num":rfqnum.rfq\_num,"split\_delvry\_date":k.split\_delvry\_date,"unique\_nbr":k.unique\_nbr,"accept\_prod\_qty":k.accept\_prod\_qty}

split\_list.append(info)

details["split\_info"]=split\_list

more\_info.append(details)

details2["rfq\_list"]=more\_info

rfq\_list.append(details2)

newlist = sorted(rfq\_list, key=lambda k: k['rfq\_num'])

#print newlist

return {"status":"Success","more\_info":newlist}

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data = request.get\_json(force=True)

rfq\_num= json\_data['rfq\_num']

status= json\_data['status']

approved\_by= json\_data['userid']

user\_data=DealerOrderData.objects(status='PO Generated',rfq\_num=rfq\_num)

if user\_data.count()==0:

return {"status":"Status already Changed"}

for prod in user\_data:

prod.status=status

prod.po\_approved\_by=approved\_by

prod.save()

num=DealerOrderData.objects.get(rfq\_num=rfq\_num)

sms\_num=Reg\_Dealer.objects.get(id=num.userid)

msg='Purchase Order No. '+num.po\_num+' has been Confirmed.'

mobile=str(sms\_num.dealer\_mobile)

sms(msg,mobile)

urldata ='Purchase Order No. '+num.po\_num+' has been Confirmed.'

sendmail(sendmail,[sms\_num.dealer\_mail,'PO Confirmed'],urldata)

return{"status":"Success"}

class DealerPODetails(Resource):#DEALER

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

user\_id=request.args['user\_id']

rfq\_list=[]

more\_info=[]

for rfq\_objects in DealerOrderData.objects(userid=user\_id):

for data in rfq\_objects.product\_info:

for k in data.split\_products:

details={}

details["productid"]=data.productid

details["modelno"]=data.modelno

details["brand"]=data.brand

details["qty"]=data.qty

details["mrp"]=data.mrp

details["total"]=data.total

details["discount"]=data.discount

details["rfq\_num"]=rfq\_objects.rfq\_num

details["accept\_productname"]=k.accept\_productname

details["accept\_prod\_qty"]=k.accept\_prod\_qty

details["balance\_qty"]=k.balance\_qty

details["split\_delvry\_date"]=k.split\_delvry\_date

details["split\_status"]=k.split\_status

details["remarks\_dealer"]=k.remarks\_dealer

details["remarks\_internal"]=k.remarks\_internal

details["product\_status"]=k.product\_status

details["unique\_nbr"]=k.unique\_nbr

details["dealer\_status"]=k.dealer\_status

more\_info.append(details)

rfq\_object=DealerOrderData.objects(userid=user\_id)

details2={}

userid=rfq\_objects.userid

dealer\_info=Reg\_Dealer.objects.get(id=userid)

details2['rfq\_num']=rfq\_object[0].rfq\_num

details2['dealer\_city']=dealer\_info.dealer\_city

details2['dealer\_name']=dealer\_info.dealer\_name

details2['dealer\_mobile']=dealer\_info.dealer\_mobile

details2['dealer\_mail']=dealer\_info.dealer\_mail

details2['rfq\_num']=rfq\_object[0].rfq\_num

details2['po\_num']=rfq\_object[0].po\_num

details2['status']=rfq\_object[0].status

details2['po\_gen\_date']=str(rfq\_object[0].po\_gen\_date)

rfq\_list.append(details2)

return {"status":"Success","rfq\_list":rfq\_list,"more\_info":more\_info}

class RfqGet(Resource):

def get(self):

rfq\_args=request.args['rfq\_num']

rfq\_info=DealerOrderData.objects.get(rfq\_num=rfq\_args)

if rfq\_info:

rfq\_list=[]

for prod in rfq\_info.product\_info:

product={}

product['productname']=prod.productname

product['productid']=prod.productid

product['modelno']=prod.modelno

product['brand']=prod.brand

product['qty']=prod.qty

product['mrp']=prod.mrp

product['discount']=prod.discount

product['spcl\_disc']=prod.spcl\_disc

product['volume\_disc']=prod.volume\_disc

product['total']=prod.total

product['savings']=prod.savings

rfq\_list.append(product)

return {"status":"Success","rfq\_list":rfq\_list}

else:

return {"status":"no records found"}

class Pending\_PO\_Get(Resource):

def get(self):

rfq\_list=[]

for rfq\_objects in DealerOrderData.objects(status='Process'):

if rfq\_objects==0:

return {"status":"no records found"}

details={}

details['rfq\_num']=rfq\_objects.rfq\_num

details['remarks']=rfq\_objects.remarks

details['rfq\_generate\_date']=str(rfq\_objects.rfq\_generate\_date)

details['sales\_manager']=rfq\_objects.sales\_manager

userid=rfq\_objects.userid

dealer\_name=Reg\_Dealer.objects.get(id=userid)

details['dealer\_name']=dealer\_name.dealer\_name

total\_value=0.0

total\_qty=0

for prod in rfq\_objects.product\_info:

product={}

product['po\_approval\_date']=str(rfq\_objects.po\_gen\_date)

product['po\_num']=rfq\_objects.po\_num

product['unique\_nbr']=prod.split\_products[0].unique\_nbr

total\_value=total\_value+float(prod.total)

print total\_value

total\_qty=total\_qty+int(prod.qty)

details['total\_value']=total\_value

details['total\_qty']=total\_qty

details['products']=product

rfq\_list.append(details)

return {"status":"Success","rfq\_list":rfq\_list}

#---------------------------------------

def synchronized(wrapped):

lock = threading.Lock()

@functools.wraps(wrapped)

def \_wrap(\*args, \*\*kwargs):

with lock:

return wrapped(\*args, \*\*kwargs)

return \_wrap

class POGenerateNo(Resource):

def appendzeros(self,package):

packagestr = ''

if package < 9:

packagestr = '0000'+str(package)

elif package < 99:

packagestr = '000'+str(package)

elif package < 999:

packagestr = '00'+str(package)

elif package < 9999:

packagestr = '0'+str(package)

else:

packagestr = str(package)

return packagestr

def getRfq(self,rfqNum):

presentRkfqNum=''

package = 0

packageStr = ''

if rfqNum.startswith('PO'):

package = int(rfqNum[3:8])

package += 1

packageStr = self.appendzeros(package)

presentRfqNum ='PO'+ packageStr

else:

packageStr = self.appendzeros(1)

presentRfqNum = 'PO'+ packageStr

return presentRfqNum

@synchronized

def get(self):

lastRfq = LatestPo.objects()

rfqReady=''

if lastRfq.count()==0:

rfqReady = self.getRfq('000000000000')

else:

rfqReady = self.getRfq(lastRfq[0].po\_num)

return rfqReady

@synchronized

def put(self, rfqCreated):

lastRfq = LatestPo.objects()

if lastRfq.count()==0:

lastRfq = LatestPo(po\_num=rfqCreated)

lastRfq.save()

else:

lastRfq = LatestPo(id=lastRfq[0].id)

lastRfq.po\_num = rfqCreated

lastRfq.save()

return "put success"

def delete(self):

lastRfq = LatestPo.objects()

lastRfq.delete()

return "success"

#---------------------------------------

def synchronized(wrapped):

lock = threading.Lock()

@functools.wraps(wrapped)

def \_wrap(\*args, \*\*kwargs):

with lock:

return wrapped(\*args, \*\*kwargs)

return \_wrap

class INVGenerateNo(Resource):

def appendzeros(self,package):

packagestr = ''

if package < 9:

packagestr = '0000'+str(package)

elif package < 99:

packagestr = '000'+str(package)

elif package < 999:

packagestr = '00'+str(package)

elif package < 9999:

packagestr = '0'+str(package)

else:

packagestr = str(package)

return packagestr

def getRfq(self,rfqNum):

presentRkfqNum=''

package = 0

packageStr = ''

if rfqNum.startswith('IN'):

package = int(rfqNum[3:8])

package += 1

packageStr = self.appendzeros(package)

presentRfqNum ='IN'+ packageStr

else:

packageStr = self.appendzeros(1)

presentRfqNum = 'IN'+ packageStr

return presentRfqNum

@synchronized

def get(self):

lastRfq = LatestInv.objects()

rfqReady=''

if lastRfq.count()==0:

rfqReady = self.getRfq('000000000000')

else:

rfqReady = self.getRfq(lastRfq[0].inv\_num)

return rfqReady

@synchronized

def put(self, rfqCreated):

lastRfq = LatestInv.objects()

if lastRfq.count()==0:

lastRfq = LatestInv(inv\_num=rfqCreated)

lastRfq.save()

else:

lastRfq = LatestInv(id=lastRfq[0].id)

lastRfq.inv\_num = rfqCreated

lastRfq.save()

return "put success"

def delete(self):

lastRfq = LatestInv.objects()

lastRfq.delete()

return "success"

#--------------------------------

class Newchalan(Resource):

def get(self):

rfq\_num=ChalanGenerateNo().get()

ChalanGenerateNo().put(rfq\_num)

return {"status":"success","orderid":rfq\_num}

#---------------------------------------

def synchronized(wrapped):

lock = threading.Lock()

@functools.wraps(wrapped)

def \_wrap(\*args, \*\*kwargs):

with lock:

return wrapped(\*args, \*\*kwargs)

return \_wrap

class ChalanGenerateNo(Resource):

def appendzeros(self,package):

packagestr = ''

if package < 9:

packagestr = '0000'+str(package)

elif package < 99:

packagestr = '000'+str(package)

elif package < 999:

packagestr = '00'+str(package)

elif package < 9999:

packagestr = '0'+str(package)

else:

packagestr = str(package)

return packagestr

def getRfq(self,rfqNum):

presentRkfqNum=''

package = 0

packageStr = ''

if rfqNum.startswith('CH'):

package = int(rfqNum[2:7])

package += 1

packageStr = self.appendzeros(package)

presentRfqNum ='CH'+ packageStr

else:

packageStr = self.appendzeros(1)

presentRfqNum = 'CH'+ packageStr

return presentRfqNum

@synchronized

def get(self):

lastRfq = LatestChalanNo.objects()

rfqReady=''

if lastRfq.count()==0:

rfqReady = self.getRfq('000000000000')

else:

rfqReady = self.getRfq(lastRfq[0].chalan\_num)

return rfqReady

@synchronized

def put(self, rfqCreated):

lastRfq = LatestChalanNo.objects()

if lastRfq.count()==0:

lastRfq = LatestChalanNo(chalan\_num=rfqCreated)

lastRfq.save()

else:

lastRfq = LatestChalanNo(id=lastRfq[0].id)

lastRfq.chalan\_num = rfqCreated

lastRfq.save()

return "put success"

def delete(self):

lastRfq = LatestChalanNo.objects()

lastRfq.delete()

return "success"

#--------------------------------

class PoGeneration(Resource):

def post(self):#split

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

statuss=json\_data['product\_status']

user\_id=json\_data['userid']

rfq\_num=json\_data['rfq\_num']

user\_check=Usertype.objects.get(id=user\_id)

for user\_data in DealerOrderData.objects(rfq\_num=rfq\_num,status='Pending',sales\_manager\_mobile=user\_check.user\_mobile):

if user\_data==0:

return {"status":"Success","pogeneration":"fail"}

if statuss=='Split':

for k in user\_data.product\_info:

k.status='Pending for Dealer side'

for acpt\_prod in json\_data['accept\_products']:

if k.productname==acpt\_prod['accept\_productname']:

accept\_productname=acpt\_prod['accept\_productname']

print accept\_productname

accept\_prod\_qty=str(acpt\_prod['accept\_prod\_qty'])

balance\_qty=str(acpt\_prod['balance\_qty'])

split\_delvry\_date=str(acpt\_prod['split\_delvry\_date'])

remarks\_dealer=acpt\_prod['remarks\_dealer']

remarks\_internal=acpt\_prod['remarks\_internal']

split\_status=acpt\_prod['split\_status']

dealer\_status='Inprogress'

product\_status='Split'

print balance\_qty,'balance\_qty1'

listofno = random.sample(set([0,1,2,3,4,5,6,7,8,9]),10)

randomnumber="".join(str(numbers) for numbers in listofno)

unique\_nbr=randomnumber

acpt\_prod\_data=Split\_Product\_Info(accept\_productname,accept\_prod\_qty,balance\_qty,split\_delvry\_date,remarks\_dealer,remarks\_internal,split\_status,product\_status,dealer\_status,unique\_nbr)

k.split\_products.append(acpt\_prod\_data)

user\_data.save()

split\_data=DealerOrderData.objects(rfq\_num=rfq\_num,sales\_manager\_mobile=user\_check.user\_mobile).filter(product\_info\_\_split\_products\_\_match={'product\_status':'Split'}).only('accept\_productname','accept\_prod\_qty','split\_status','split\_delvry\_date')

data\_list=json.loads(split\_data.to\_json())

print data\_list

return {"status":"Success","split\_data":data\_list}

if statuss=='Accept':

for k in user\_data.product\_info:

k.status='Pending for Dealer side'

for acpt\_prod in json\_data['accept\_products']:

if k.productname==acpt\_prod['accept\_productname']:

accept\_productname=acpt\_prod['accept\_productname']

accept\_prod\_qty=str(acpt\_prod['accept\_prod\_qty'])

balance\_qty=''

split\_delvry\_date=str(acpt\_prod['split\_delvry\_date'])

remarks\_dealer=acpt\_prod['remarks\_dealer']

remarks\_internal=acpt\_prod['remarks\_internal']

split\_status=acpt\_prod['split\_status']

dealer\_status='Inprogress'

product\_status='Accept'

acpt\_prod\_data=Split\_Product\_Info(accept\_productname,accept\_prod\_qty,balance\_qty,split\_delvry\_date,remarks\_dealer,remarks\_internal,split\_status,product\_status,dealer\_status)

k.split\_products.append(acpt\_prod\_data)

user\_data.save()

return {"status":"Success"}

if statuss=='Reject':

for k in user\_data.product\_info:

k.status='Reject'

for acpt\_prod in json\_data['accept\_products']:

if k.productname==acpt\_prod['accept\_productname']:

accept\_productname=acpt\_prod['accept\_productname']

split\_delvry\_date=str(acpt\_prod['split\_delvry\_date'])

accept\_prod\_qty=str(acpt\_prod['accept\_prod\_qty'])

balance\_qty=''

split\_delvry\_date=''

remarks\_dealer=acpt\_prod['remarks\_dealer']

remarks\_internal=acpt\_prod['remarks\_internal']

split\_status=acpt\_prod['split\_status']

dealer\_status='Reject'

product\_status='Reject'

acpt\_prod\_data=Split\_Product\_Info(accept\_productname,accept\_prod\_qty,balance\_qty,split\_delvry\_date,remarks\_dealer,remarks\_internal,split\_status,product\_status,dealer\_status)

k.split\_products.append(acpt\_prod\_data)

user\_data.save()

return {"status":"Success","postatus":"po generation rejected."}

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

rfq\_args=request.args['rfq\_num']

deal\_check=DealerOrderData.objects(rfq\_num=rfq\_args)

if deal\_check.count()==0:

return {"status":"no records found"}

for rfq\_objects in DealerOrderData.objects(rfq\_num=rfq\_args):

product\_list=[]

total\_value=0

total\_qty=0

for prod in rfq\_objects.product\_info:

wh\_qty=ProdinvWH.objects(prod\_desc=prod.productname)

product={}

product['productname']=prod.productname

product['productid']=prod.productid

product['modelno']=prod.modelno

product['dealer\_qty']=prod.qty

product['wh\_quantity']=wh\_qty[0].quantity

product['product\_status']=prod.status

if prod.status=='Pending' or len(prod.split\_products)==0:

product["product\_status"]='Pending for SM'

product['balance\_qty']=prod.qty

for info in prod.split\_products:

product['accept\_prod\_qty']=info.accept\_prod\_qty

if info.product\_status=="Split":

product['unique\_nbr']=info.unique\_nbr

product['balance\_qty']=info.balance\_qty#int(info.balance\_qty)-int(info.accept\_prod\_qty)

if info.product\_status!="Split":

product['balance\_qty']=int(prod.qty)-int(info.accept\_prod\_qty)

product['split\_balance\_qty']=info.balance\_qty

product['split\_delvry\_date']=info.split\_delvry\_date

product['remarks\_dealer']=info.remarks\_dealer

product['remarks\_internal']=info.remarks\_internal

product['product\_status']=info.product\_status

product\_list.append(product)

return {"status":"Success","rfq\_list":product\_list,"User\_id":rfq\_objects.userid,"rfq\_num":rfq\_objects.rfq\_num}

def put(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

#name=json\_data['product\_name']

user\_id=json\_data['userid']

rfq\_num=json\_data['rfq\_num']

user\_check=Usertype.objects.get(id=user\_id)

for user\_data in DealerOrderData.objects(rfq\_num=rfq\_num,status='Pending',sales\_manager\_mobile=user\_check.user\_mobile):

if user\_data==0:

return {"status":"Success","pogeneration":"fail"}

for k in user\_data.product\_info:

print k.productname

for acpt\_prod in json\_data['accept\_products']:

accept\_productname=acpt\_prod['accept\_productname']

accept\_prod\_qty=str(acpt\_prod['accept\_prod\_qty'])

balance\_qty=acpt\_prod['balance\_qty']

print accept\_prod\_qty,'accept\_prod\_qty'

print balance\_qty,'balance\_qty'

split\_delvry\_date=str(acpt\_prod['split\_delvry\_date'])

remarks\_dealer=acpt\_prod['remarks\_dealer']

remarks\_internal=acpt\_prod['remarks\_internal']

split\_status=acpt\_prod['split\_status']

dealer\_status=acpt\_prod['dealer\_status']

product\_status=acpt\_prod['product\_status']

if k.productname==acpt\_prod['accept\_productname'] and product\_status=='Split':

listofno = random.sample(set([0,1,2,3,4,5,6,7,8,9]),10)

randomnumber="".join(str(numbers) for numbers in listofno)

bal\_qty=int(balance\_qty)-int(accept\_prod\_qty)

acpt\_prod\_data1=Split\_Product\_Info(accept\_productname,accept\_prod\_qty,str(balance\_qty),split\_delvry\_date,remarks\_dealer,remarks\_internal,split\_status,product\_status,dealer\_status,randomnumber)

k.split\_products.append(acpt\_prod\_data1)

k.save()

user\_data.save()

print k.split\_products

if k.productname==acpt\_prod['accept\_productname'] and product\_status=='Accept':

del k.split\_products[:]

k.save()

user\_data.save()

print k.split\_products,"k.split\_products"

acpt\_prod\_data=Split\_Product\_Info(accept\_productname,accept\_prod\_qty,accept\_prod\_qty,split\_delvry\_date,remarks\_dealer,remarks\_internal,split\_status,product\_status,dealer\_status)

k.split\_products.append(acpt\_prod\_data)

k.status='Accept'

user\_data.save()

if k.productname==acpt\_prod['accept\_productname'] and product\_status=='Reject':

del k.split\_products[:]

k.save()

user\_data.save()

acpt\_prod\_data=Split\_Product\_Info(accept\_productname,accept\_prod\_qty,accept\_prod\_qty,split\_delvry\_date,remarks\_dealer,remarks\_internal,split\_status,product\_status,dealer\_status)

k.split\_products.append(acpt\_prod\_data)

k.status='Reject'

user\_data.save()

split\_data=DealerOrderData.objects(rfq\_num=rfq\_num,sales\_manager\_mobile=user\_check.user\_mobile).filter(product\_info\_\_split\_products\_\_match={'product\_status':'Split'}).only('accept\_productname','accept\_prod\_qty','product\_status','split\_delvry\_date')

data\_list=json.loads(split\_data.to\_json())

return {"status":"Success","split\_data":data\_list}

class SplitData(Resource):

def get(self):#split

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

userid=request.args['user\_id']

rfq\_num=request.args['rfq\_num']

user\_check=Usertype.objects.get(id=userid)

if user\_check>0:

for rfq\_objects in DealerOrderData.objects(sales\_manager\_mobile=user\_check.user\_mobile,rfq\_num=rfq\_num):

data\_list=[]

for data in rfq\_objects.product\_info:

for k in data.split\_products:

details={}

details["productid"]=data.productid

details["accept\_productname"]=k.accept\_productname

details["accept\_prod\_qty"]=k.accept\_prod\_qty

details["balance\_qty"]=k.balance\_qty

details["split\_delvry\_date"]=k.split\_delvry\_date

details["split\_status"]=k.split\_status

details["unique\_nbr"]=k.unique\_nbr

details["remarks\_dealer"]=k.remarks\_dealer

details["remarks\_internal"]=k.remarks\_internal

details["product\_status"]=k.product\_status

print k.product\_status,' ',k.accept\_productname

main\_data=json.loads(json.dumps(details))

data\_list.append(main\_data)

return {"status":"Success","rfq\_list":data\_list}

for rfq\_objects in DealerOrderData.objects(userid=userid,rfq\_num=rfq\_num):

data\_list=[]

for data in rfq\_objects.product\_info:

for k in data.split\_products:

details={}

details["productid"]=data.productid

details["accept\_productname"]=k.accept\_productname

details["accept\_prod\_qty"]=k.accept\_prod\_qty

details["balance\_qty"]=k.balance\_qty

details["unique\_nbr"]=k.unique\_nbr

details["split\_delvry\_date"]=k.split\_delvry\_date

details["split\_status"]=k.split\_status

details["remarks\_dealer"]=k.remarks\_dealer

details["remarks\_internal"]=k.remarks\_internal

details["product\_status"]=k.product\_status

main\_data=json.loads(json.dumps(details))

data\_list.append(main\_data)

return {"status":"Success","rfq\_list":data\_list}

class User\_Saving(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

usertype=json\_data['usertype']

username=json\_data['username']

user\_mail=json\_data['user\_email']

user\_mobile=json\_data['user\_mobile']

user\_password=json\_data['user\_password']

user\_saving=Usertype.objects(usertype=usertype,user\_mail=user\_mail)

if user\_saving.count()==0:

saving\_data=Usertype(usertype=usertype,username=username,user\_mail=user\_mail,user\_mobile=user\_mobile,user\_password=generate\_password\_hash(user\_password)).save()

return {"status":"Success","data":"Data saved successfully."}

else:

return {"status":"Success","Data":"User already existed."}

class Users\_list(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

users\_list=[]

for users\_objects in Usertype.objects():

users\_data=json.loads(users\_objects.to\_json())

users\_list.append(users\_data)

return {"status":"Success","Users\_info":users\_list}

class UsersDelete(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

userid=request.args['userid']

user = Usertype.objects.get(id=userid)

user.delete()

return {"status":"success","message":"record deleted successfully"}

else:

return {"status":"invalid secret key"}

#--------------------------Account Manager-----------------------------------

class PoInformation(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

userid=request.args['userid']

usertype=request.args['usertype']

print usertype

if usertype=='salesManager' :

rfq\_list=[]

sales\_name = Usertype.objects.get(usertype='salesManager',id=userid)

for rfq\_objects in DealerOrderData.objects(sales\_manager\_mobile=sales\_name.user\_mobile,status="Approved").filter(product\_info\_\_split\_products\_\_match={'dealer\_status':'Confirm'}):

dealer\_info=Reg\_Dealer.objects.get(id=rfq\_objects.userid)

details1={}

details1["dealer\_name"]=dealer\_info.dealer\_name

details1["dealer\_city"]=dealer\_info.dealer\_city

details1["dealer\_mail"]=dealer\_info.dealer\_mail

details1["dealer\_mobile"]=dealer\_info.dealer\_mobile

details1['rfq\_num']=rfq\_objects.rfq\_num

details1['status']=rfq\_objects.status

details1['po\_num']=rfq\_objects.po\_num

details1['total']=rfq\_objects.product\_info[0].total

details1['rfq\_generate\_date']=(rfq\_objects.rfq\_generate\_date).strftime("%d-%m-%Y")

details1['po\_gen\_date']=rfq\_objects.po\_gen\_date

more\_info=[]

for data in rfq\_objects.product\_info:

details={}

if data.status=='Pending':

details["product\_status"]='Pending for PO'

split\_list=[]

for k in data.split\_products:

print k

if data.productname==k.accept\_productname and k.split\_status=='Accept':

details["productid"]=data.productid

details["productname"]=data.productname

details["modelno"]=data.modelno

details["brand"]=data.brand

details["qty"]=data.qty

details["remain\_qty"]=data.remain\_qty

details["mrp"]=data.mrp

details["total"]=data.total

details["discount"]=data.discount

details["rfq\_num"]=rfq\_objects.rfq\_num

details["volume\_disc"]=data.volume\_disc

details["savings"]=data.savings

details["spcl\_disc"]=data.spcl\_disc

details["accept\_productname"]=k.accept\_productname

details["balance\_qty"]=k.balance\_qty

details["delivry\_date"]=k.split\_delvry\_date

details["split\_status"]=k.split\_status

details["product\_status"]=k.product\_status

details["dealer\_status"]=k.dealer\_status

details["rfq\_num"]=rfq\_objects.rfq\_num

details["delvry\_date"]=k.split\_delvry\_date

if data.productname==k.accept\_productname and k.product\_status=='Split':

info={"remarks\_dealer":k.remarks\_dealer,"productname":k.accept\_productname,"remarks\_internal":k.remarks\_internal,"rfq\_num":rfq\_objects.rfq\_num,"split\_delvry\_date":k.split\_delvry\_date,"unique\_nbr":k.unique\_nbr,"accept\_prod\_qty":k.accept\_prod\_qty}

split\_list.append(info)

details["split\_info"]=split\_list

more\_info.append(details)

details1["rfq\_list"]=more\_info

rfq\_list.append(details1)

newlist = sorted(rfq\_list, key=lambda k: k['rfq\_num'])

return {"status":"Success","more\_info":newlist}

if usertype=='accountant' or usertype=='whManager':

rfq\_list=[]

details2={}

checking\_data=DealerOrderData.objects(Q(status='Approved') | Q(status='Partial Inv Generated')).count()

if checking\_data==0:

return {"status":"no records found"}

if checking\_data>0:

for rfqnum in DealerOrderData.objects(Q(status='Approved') | Q(status='Partial Inv Generated')):

sales\_name=Usertype.objects.get(usertype='salesManager',user\_mobile=rfqnum.sales\_manager\_mobile)

acnt\_name=Usertype.objects.get(id=userid)

dealer\_info=Reg\_Dealer.objects.get(id=rfqnum.userid)

details2['sales\_manager']=sales\_name.username

details2['dealer\_name']=dealer\_info.dealer\_name

for bill in dealer\_info.billing\_address:

bill\_addr={}

bill\_addr['d\_no']=bill.d\_no

bill\_addr['street']=bill.street

bill\_addr['city']=bill.city

bill\_addr['state']=bill.state

bill\_addr['zipcode']=bill.zipcode

ship\_list=[]

for ship in dealer\_info.shipping\_address:

ship\_addr={}

ship\_addr['d\_no']=ship.d\_no

ship\_addr['street']=ship.street

ship\_addr['city']=ship.city

ship\_addr['state']=ship.state

ship\_addr['zipcode']=ship.zipcode

ship\_list.append(ship\_addr)

details2['billing\_address']=bill\_addr

details2['shipping\_address']=ship\_list

details2['dealer\_mail']=dealer\_info.dealer\_mail

details2['dealer\_mobile']=dealer\_info.dealer\_mobile

details2['dealer\_city']=dealer\_info.dealer\_city

details2['acnt\_mail']=acnt\_name.user\_mail

details2['acnt\_mobile']=acnt\_name.user\_mobile

details2['acnt\_name']=acnt\_name.username

details2['sales\_manager\_mobile']=sales\_name.user\_mobile

details2['rfq\_num']=rfqnum.rfq\_num

details2['status']=rfqnum.status

details2['po\_num']=rfqnum.po\_num

details2['status']=rfqnum.status

details2['po\_gen\_date']=rfqnum.po\_gen\_date

details2['sales\_manager\_mobile']=rfqnum.sales\_manager\_mobile

details2['rfq\_generate\_date']=(rfqnum.rfq\_generate\_date).strftime("%d-%m-%Y")

details2['remarks']=rfqnum.remarks

more\_info=[]

for data in rfqnum.product\_info:

details={}

if data.status=='Pending':

details["product\_status"]='Pending for PO'

split\_list=[]

for k in data.split\_products:

if data.productname==k.accept\_productname and k.split\_status=='Accept' and int(data.remain\_qty)>0:

sup\_upload=Sup\_Upload1.objects.get(upload\_id=data.productid)

details["productid"]=data.productid

details["productname"]=data.productname

details["modelno"]=data.modelno

details["brand"]=data.brand

details["pieceperCarton"]=sup\_upload.upload\_pieceperCarton

details["qty"]=data.qty

details["remain\_qty"]=data.remain\_qty

details["mrp"]=data.mrp

details["total"]=data.total

details["discount"]=data.discount

details["rfq\_num"]=rfqnum.rfq\_num

details["volume\_disc"]=data.volume\_disc

details["savings"]=data.savings

details["spcl\_disc"]=data.spcl\_disc

details["accept\_productname"]=k.accept\_productname

details["balance\_qty"]=k.balance\_qty

details["split\_status"]=k.split\_status

details["delivry\_date"]=k.split\_delvry\_date

details["product\_status"]=k.product\_status

details["dealer\_status"]=k.dealer\_status

details["rfq\_num"]=rfqnum.rfq\_num

details["delvry\_date"]=k.split\_delvry\_date

if data.productname==k.accept\_productname and k.product\_status=='Split' and rfqnum.status=='Partial Inv Generated':

for invoice\_info in Invoice\_Info.objects(po\_num=rfqnum.po\_num):

if invoice\_info.invoice\_status=='Partial Inv Generated' or invoice\_info.moreinv\_info[0].split\_inv\_info[0].inv\_status=='Partial Inv Generated':

info={}

qty=0

for inv in invoice\_info.moreinv\_info:

if inv.productname==k.accept\_productname and int(inv.remain\_qty)>0 and inv.split\_inv\_info[0].unique\_num!=0:

qty=qty+int(inv.split\_inv\_info[0].split\_inv\_qty)

details["remain\_qty"]=qty

if inv.productname==k.accept\_productname and inv.split\_inv\_info[0].unique\_num==k.unique\_nbr:

qty\_check=int(inv.split\_inv\_info[0].acpt\_inv\_qty)-int(inv.split\_inv\_info[0].p\_inv\_qty)

if qty\_check>0 and rfqnum.status=='Partial Inv Generated':

info["remarks\_dealer"]=k.remarks\_dealer

info["remarks\_internal"]=k.remarks\_internal

info["productname"]=inv.productname

info["rfq\_num"]=rfqnum.rfq\_num

info["remain\_qty"]=qty\_check

info["split\_delvry\_date"]=inv.split\_inv\_info[0].delivery\_date

info["unique\_nbr"]=inv.split\_inv\_info[0].unique\_num

info["accept\_prod\_qty"]=inv.split\_inv\_info[0].acpt\_inv\_qty

info["remain\_accept\_qty"]=inv.split\_inv\_info[0].split\_inv\_qty

split\_list.append(info)

details["split\_info"]=split\_list

if data.productname==k.accept\_productname and k.product\_status=='Split' and rfqnum.status=='Approved':

info={"remarks\_dealer":k.remarks\_dealer,"remarks\_internal":k.remarks\_internal,"productname":k.accept\_productname,"rfq\_num":rfqnum.rfq\_num,"split\_delvry\_date":k.split\_delvry\_date,"unique\_nbr":k.unique\_nbr,"accept\_prod\_qty":k.accept\_prod\_qty,"remain\_accept\_qty":k.accept\_prod\_qty}

split\_list.append(info)

details["split\_info"]=split\_list

more\_info.append(details)

details2["rfq\_list"]=more\_info

rfq\_list.append(details2)

newlist = sorted(rfq\_list, key=lambda k: k['rfq\_num'])

return {"status":"Success","more\_info":newlist}

if usertype=='Dealer':

rfq\_list=[]

for rfqnum in DealerOrderData.objects(userid=userid,status="Approved"):

sales\_name=Usertype.objects.get(usertype='salesManager',user\_mobile=rfqnum.sales\_manager\_mobile)

details2={}

details2['sales\_manager']=sales\_name.username

details2['sales\_manager\_mobile']=sales\_name.user\_mobile

details2['rfq\_num']=rfqnum.rfq\_num

details2['status']=rfqnum.status

details2['po\_num']=rfqnum.po\_num

details2['status']=rfqnum.status

details2['po\_gen\_date']=rfqnum.po\_gen\_date

details2['sales\_manager\_mobile']=rfqnum.sales\_manager\_mobile

details2['rfq\_generate\_date']=(rfqnum.rfq\_generate\_date).strftime("%d-%m-%Y")

details2['remarks']=rfqnum.remarks

more\_info=[]

for data in rfqnum.product\_info:

details={}

if data.status=='Pending':

details["product\_status"]='Pending for PO'

split\_list=[]

for k in data.split\_products:

if data.productname==k.accept\_productname and k.split\_status=='Accept':

details["productid"]=data.productid

details["productname"]=data.productname

details["modelno"]=data.modelno

details["brand"]=data.brand

details["qty"]=data.qty

details["remain\_qty"]=data.remain\_qty

details["mrp"]=data.mrp

details["total"]=data.total

details["discount"]=data.discount

details["rfq\_num"]=rfqnum.rfq\_num

details["volume\_disc"]=data.volume\_disc

details["savings"]=data.savings

details["spcl\_disc"]=data.spcl\_disc

details["accept\_productname"]=k.accept\_productname

details["balance\_qty"]=k.balance\_qty

details["delivry\_date"]=k.split\_delvry\_date

details["split\_status"]=k.split\_status

details["product\_status"]=k.product\_status

details["dealer\_status"]=k.dealer\_status

details["rfq\_num"]=rfqnum.rfq\_num

details["delvry\_date"]=k.split\_delvry\_date

if data.productname==k.accept\_productname and k.product\_status=='Split':

info={"remarks\_dealer":k.remarks\_dealer,"remarks\_internal":k.remarks\_internal,"productname":k.accept\_productname,"rfq\_num":rfqnum.rfq\_num,"split\_delvry\_date":k.split\_delvry\_date,"unique\_nbr":k.unique\_nbr,"accept\_prod\_qty":k.accept\_prod\_qty}

split\_list.append(info)

details["split\_info"]=split\_list

more\_info.append(details)

details2["rfq\_list"]=more\_info

rfq\_list.append(details2)

newlist = sorted(rfq\_list, key=lambda k: k['rfq\_num'])

#print newlist

return {"status":"Success","more\_info":newlist}

'''

class InvGeneration(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

po\_num=json\_data['po\_num']

userid=json\_data['user\_id']

invoice\_date=json\_data['invoice\_date'] if 'invoice\_date' in json\_data else ''

inv\_num=INVGenerateNo().get()

#invoice\_data=Invoice\_Info.objects(po\_num=po\_num,userid=userid)

invoice\_info=Invoice\_Info(po\_num=po\_num,userid=userid,p\_inv\_num='P'+inv\_num,invoice\_date=str(invoice\_date),chalan\_num='',chalan\_date='',delivery\_type='',driver\_name='',dl\_num='',driver\_contact='',vehicle\_num='',remarks='',courier\_num='',courier\_name='',courier\_boy='',weight='').save()

for invoice in json\_data['invoice\_info']:

total\_amt=str(invoice['total\_amt'])

product\_qty=str(invoice['product\_qty'])

remain\_qty=str(invoice['rem\_qty'])

inv\_qty=str(invoice['inv\_qty'])

acpt\_inv\_qty=str(invoice['acpt\_inv\_qty'])

split\_inv\_qty=str(invoice['split\_inv\_qty'])

productid=str(invoice['product\_id'])

productname=str(invoice['product\_name'])

unic\_num=str(invoice['unique\_nbr'])

delivry\_date=str(invoice['delivry\_date'])

product\_status=invoice['product\_status']

piece\_per\_carton=invoice['piece\_per\_carton']

print piece\_per\_carton,'piece\_per\_carton'

if product\_status=='Accept':

status\_update=DealerOrderData.objects.get(po\_num=po\_num)

for prod in status\_update.product\_info:

if prod.productname==productname:

prod.remain\_qty=str(invoice['rem\_qty'])

prod.save()

status\_update.save()

split\_inv1=SplitInv(inv\_qty,'',acpt\_inv\_qty,'','','',unic\_num,split\_inv\_qty,delivry\_date)

inv\_data1=IvoiceMoreInfo(total\_amt=total\_amt,req\_product\_qty=product\_qty,remain\_qty=remain\_qty,piece\_per\_carton=piece\_per\_carton,avl\_carton='',productid=productid,productname=productname)

inv\_data1.split\_inv\_info.append(split\_inv1)

invoice\_info.moreinv\_info.append(inv\_data1)

invoice\_info.save()

if product\_status=='Split':

status\_update=DealerOrderData.objects.get(po\_num=po\_num)

for prod in status\_update.product\_info:

if prod.productname==productname:

print prod.remain\_qty,'prod.remain\_qty'

print inv\_qty,"invoice['rem\_qty']"

qty\_update=int(prod.remain\_qty)-int(inv\_qty)

prod.remain\_qty=str(qty\_update)

print prod.remain\_qty,'prod.remain\_qty'

prod.save()

status\_update.save()

split\_inv1=SplitInv(inv\_qty,'',acpt\_inv\_qty,'','','',unic\_num,split\_inv\_qty,delivry\_date)

inv\_data1=IvoiceMoreInfo(total\_amt=total\_amt,req\_product\_qty=product\_qty,remain\_qty=remain\_qty,piece\_per\_carton=piece\_per\_carton,avl\_carton='',productid=productid,productname=productname)

inv\_data1.split\_inv\_info.append(split\_inv1)

invoice\_info.moreinv\_info.append(inv\_data1)

invoice\_info.save()

inv\_data=Invoice\_Info.objects.get(po\_num=po\_num,userid=userid,p\_inv\_num='P'+inv\_num)

totalqty=sum(int(prod['remain\_qty']) for prod in inv\_data.moreinv\_info)

total\_qty=int(totalqty)

print total\_qty,'total\_qty'

if total\_qty>0:

status\_update=DealerOrderData.objects.get(po\_num=po\_num)

status\_update.status='Partial Inv Generated'

status\_update.save()

for i in inv\_data.moreinv\_info:

for j in i.split\_inv\_info:

if int(i.remain\_qty)>0:

inv\_data.invoice\_status='Partial Inv Generated'

j.inv\_status='Partial Inv Generated'

j.save()

if int(i.remain\_qty)==0:

inv\_data.invoice\_status='PInv Generated'

j.inv\_status='PInv Generated'

j.save()

inv\_data.save()

if total\_qty==0:

status\_update=DealerOrderData.objects.get(po\_num=po\_num)

status\_update.status='PInv Generated'

status\_update.save()

for i in inv\_data.moreinv\_info:

for j in i.split\_inv\_info:

if int(i.remain\_qty)>0:

inv\_data.invoice\_status='Partial Inv Generated'

j.inv\_status='Partial Inv Generated'

j.save()

if int(i.remain\_qty)==0:

inv\_data.invoice\_status='PInv Generated'

j.inv\_status='PInv Generated'

j.save()

inv\_data.save()

INVGenerateNo().put(inv\_num)

return {"status":"Succes","inv\_num":'P'+inv\_num}

'''

class InvGeneration(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

po\_num=json\_data['po\_num']

userid=json\_data['user\_id']

invoice\_date=json\_data['invoice\_date'] if 'invoice\_date' in json\_data else ''

inv\_num=INVGenerateNo().get()

invoice\_info=Invoice\_Info(po\_num=po\_num,userid=userid,p\_inv\_num='P'+inv\_num,invoice\_date=str(invoice\_date),chalan\_num='',chalan\_date='',delivery\_type='',driver\_name='',dl\_num='',driver\_contact='',vehicle\_num='',remarks='',courier\_num='',courier\_name='',courier\_boy='',weight='').save()

for invoice in json\_data['invoice\_info']:

total\_amt=str(invoice['total\_amt'])

product\_qty=str(invoice['product\_qty'])

remain\_qty=str(invoice['rem\_qty'])

inv\_qty=str(invoice['inv\_qty'])

acpt\_inv\_qty=str(invoice['acpt\_inv\_qty'])

split\_inv\_qty=str(invoice['split\_inv\_qty'])

productid=str(invoice['product\_id'])

productname=str(invoice['product\_name'])

unic\_num=str(invoice['unique\_nbr'])

delivry\_date=str(invoice['delivry\_date'])

piece\_per\_carton=invoice['piece\_per\_carton']

status\_update=DealerOrderData.objects.get(po\_num=po\_num)

for prod in status\_update.product\_info:

if prod.productname==productname:

prod.remain\_qty=str(remain\_qty)

prod.save()

status\_update.save()

split\_inv1=SplitInv(inv\_qty,'',acpt\_inv\_qty,'','','',unic\_num,split\_inv\_qty,delivry\_date)

inv\_data1=IvoiceMoreInfo(total\_amt=total\_amt,req\_product\_qty=product\_qty,remain\_qty=remain\_qty,piece\_per\_carton=piece\_per\_carton,avl\_carton='',productid=productid,productname=productname)

inv\_data1.split\_inv\_info.append(split\_inv1)

invoice\_info.moreinv\_info.append(inv\_data1)

invoice\_info.save()

inv\_data=Invoice\_Info.objects.get(po\_num=po\_num,userid=userid,p\_inv\_num='P'+inv\_num)

totalqty=sum(int(prod['remain\_qty']) for prod in inv\_data.moreinv\_info)

total\_qty=int(totalqty)

if total\_qty>0:

status\_update=DealerOrderData.objects.get(po\_num=po\_num)

status\_update.status='Partial Inv Generated'

status\_update.save()

if total\_qty==0:

status\_update=DealerOrderData.objects.get(po\_num=po\_num)

status\_update.status='PInv Generated'

status\_update.save()

for i in inv\_data.moreinv\_info:

for j in i.split\_inv\_info:

if int(i.remain\_qty)>0:

inv\_data.invoice\_status='Partial Inv Generated'

j.inv\_status='Partial Inv Generated'

j.save()

if int(i.remain\_qty)==0:

inv\_data.invoice\_status='PInv Generated'

j.inv\_status='PInv Generated'

j.save()

inv\_data.save()

INVGenerateNo().put(inv\_num)

return {"status":"Succes","inv\_num":'P'+inv\_num}

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

inv\_list=[]

for inv\_details in Invoice\_Info.objects(invoice\_status='PInv Generated'):

inv\_data=json.loads(inv\_details.to\_json())

inv\_list.append(inv\_data)

return {"status":"Success","inv\_info":inv\_list}

def put(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

po\_num=json\_data['po\_num']

for items in json\_data['acpt\_carton\_info']:

inv\_num=INVGenerateNo().get()

prod\_name=items['product\_name']

p\_inv\_no=items['inv\_num']

unique\_num=items['unique\_num']

acpt\_qty=items['inv\_acc\_qty']

totl\_amt=items['total\_amt']

remain\_qty=items['remain\_qty']

deal\_check=DealerOrderData.objects.get(po\_num=po\_num)

po\_check=Invoice\_Info.objects.get(po\_num=po\_num,p\_inv\_num=p\_inv\_no)

for prod in deal\_check.product\_info:

for more in po\_check.moreinv\_info:

for inv in more.split\_inv\_info:

if more.productname==prod\_name and inv.unique\_num==unique\_num and prod.productname==prod\_name:

prod.remain\_qty=str(remain\_qty)

more.remain\_qty=str(remain\_qty)

more.total\_amt=str(totl\_amt)

po\_check.f\_inv\_num='F'+str(inv\_num)

inv.acpt\_carton=str(acpt\_qty)

po\_check.invoice\_status='Final Inv Generated'

more.save()

prod.save()

inv.save()

if more.productname==prod\_name and inv.unique\_num=='0' and prod.productname==prod\_name:

prod.remain\_qty=str(remain\_qty)

more.remain\_qty=str(remain\_qty)

more.total\_amt=str(totl\_amt)

po\_check.f\_inv\_num='F'+str(inv\_num)

inv.acpt\_carton=str(acpt\_qty)

po\_check.invoice\_status='Final Inv Generated'

more.save()

prod.save()

inv.save()

if int(prod.remain\_qty)==0:

deal\_check.status='Final Inv Generated'

if int(more.remain\_qty)==0:

inv.inv\_status='Final Inv Generated'

deal\_check.save()

inv.save()

po\_check.save()

num=DealerOrderData.objects.get(po\_num=po\_num)

sms\_num=Reg\_Dealer.objects.get(id=num.userid)

msg='Invoice No. '+inv\_num+' generated for your order.'

mobile=str(sms\_num.dealer\_mobile)

sms(msg,mobile)

urldata ='Sales Manager Confirmed Your Purchase Order No.'+po\_num+'.'

sendmail(sendmail,[sms\_num.dealer\_mail,'Invoice Generated'],urldata)

INVGenerateNo().put(inv\_num)

return {"status":"qty updated successfully.","inv\_num":inv\_num}

else:

return {"status":"Invalid secret-key"}

class InvChalan(Resource):

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

po\_num=json\_data['po\_num']

inv\_num=json\_data['inv\_num']

chalan\_num=ChalanGenerateNo().get()

invoice\_check=Invoice\_Info.objects(po\_num=po\_num,f\_inv\_num=inv\_num)

if invoice\_check.count()==0:

return {"status":"no records found"}

chalan\_num=ChalanGenerateNo().get()

invoice\_data=Invoice\_Info.objects.get(po\_num=po\_num,f\_inv\_num=inv\_num)

invoice\_data.chalan\_num=chalan\_num

invoice\_data.chalan\_date=str(datetime.datetime.now().date())

invoice\_data.chalan\_status='Ch-Generated'

for inv in invoice\_data.moreinv\_info:

for split in inv.split\_inv\_info:

for invoice in json\_data['invoice\_info']:

if invoice['delivery\_type']=='Vehicle':

invoice\_data.delivery\_type=invoice['delivery\_type']

invoice\_data.driver\_name=str(invoice['driver\_name'])

invoice\_data.dl\_num=str(invoice['dl\_num'])#licence num

invoice\_data.driver\_contact=str(invoice['driver\_contact'])

invoice\_data.remarks=str(invoice['remarks'])

split.remarks=str(invoice['remarks'])

invoice\_data.vehicle\_num=str(invoice['vehicle\_num'])

split.save()

ChalanGenerateNo().put(chalan\_num)

if invoice['delivery\_type']=='Courier':

invoice\_data.delivery\_type=invoice['delivery\_type']

invoice\_data.courier\_num=str(invoice['courier\_num'])

invoice\_data.courier\_name=str(invoice['courier\_name'])

invoice\_data.courier\_boy=str(invoice['courier\_boy'])

invoice\_data.remarks=str(invoice['remarks'])

split.remarks=str(invoice['remarks'])

invoice\_data.weight=str(invoice['weight'])

split.save()

invoice\_data.save()

ChalanGenerateNo().put(chalan\_num)

return {"status":"Succes","chalan\_num":chalan\_num}

def get(self):

chalan\_nbr=request.args['chalan\_num']

chalan\_data=Invoice\_Info.objects(chalan\_num=chalan\_nbr).only('po\_num','chalan\_status','f\_inv\_num','p\_inv\_num','invoice\_date','invoice\_status','moreinv\_info\_\_avl\_carton','moreinv\_info\_\_piece\_per\_carton','moreinv\_info\_\_total\_amt','moreinv\_info\_\_productname','moreinv\_info\_\_req\_product\_qty','moreinv\_info\_\_invoice\_status','moreinv\_info\_\_remain\_qty','delivery\_type','moreinv\_info\_\_split\_inv\_info\_\_delivery\_date','moreinv\_info\_\_split\_inv\_info\_\_unique\_num','moreinv\_info\_\_split\_inv\_info\_\_acpt\_inv\_qty','moreinv\_info\_\_split\_inv\_info\_\_p\_inv\_qty','moreinv\_info\_\_split\_inv\_info\_\_f\_inv\_num','chalan\_num','driver\_name','dl\_num','driver\_contact','vehicle\_num','courier\_num','courier\_name','courier\_boy','weight','moreinv\_info\_\_split\_inv\_info\_\_inv\_status','moreinv\_info\_\_split\_inv\_info\_\_acpt\_carton')

chalan\_list=json.loads(chalan\_data.to\_json())

if chalan\_data:

return {"status":"success","chalan\_list":chalan\_list}

else:

return {"status":"chalan doesnot exists"}

class Estmate\_Avlqty\_Update(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

usertype=request.args['usertype']

#po\_num=request.args['po\_num']

if usertype=='whManager':

chalan\_check= Invoice\_Info.objects(Q(invoice\_status='PInv Generated') | Q(invoice\_status='Partial Inv Generated')).count()

print chalan\_check,'chalan\_check'

if chalan\_check >0:

est\_qty= Invoice\_Info.objects(Q(invoice\_status='PInv Generated') | Q(invoice\_status='Partial Inv Generated')).only('po\_num','invoice\_date','p\_inv\_num','invoice\_status','moreinv\_info\_\_avl\_carton','moreinv\_info\_\_piece\_per\_carton','moreinv\_info\_\_total\_amt','moreinv\_info\_\_productname','moreinv\_info\_\_req\_product\_qty','moreinv\_info\_\_invoice\_status','moreinv\_info\_\_remain\_qty','moreinv\_info\_\_split\_inv\_info\_\_delivery\_date','moreinv\_info\_\_split\_inv\_info\_\_acpt\_inv\_qty','moreinv\_info\_\_split\_inv\_info\_\_p\_inv\_qty','chalan\_num','moreinv\_info\_\_split\_inv\_info\_\_inv\_status','moreinv\_info\_\_split\_inv\_info\_\_unique\_num')

est\_qty\_info=json.loads(est\_qty.to\_json())

for info in est\_qty\_info:

est\_qty\_list=[]

for data in info['moreinv\_info']:

for detail in data['split\_inv\_info']:

details1={}

if int(detail['p\_inv\_qty'])>0:

details1['p\_inv\_qty']=detail['p\_inv\_qty']

details1['total\_amt']=format(float(data['total\_amt']),'.2f')

amt=data['total\_amt']

details1['req\_product\_qty']=data['req\_product\_qty']

details1['remain\_qty']=data['remain\_qty']

details1['pieceperCorton']=data['piece\_per\_carton']

details1['productname']=data['productname']

details1['unique\_num']=detail['unique\_num']

details1["estimated\_pieceperCarton"]=int(data['req\_product\_qty'])/int(data['piece\_per\_carton'])

details1['avl\_qty']=data['avl\_carton']

details1['acpt\_inv\_qty']=detail['acpt\_inv\_qty']

details1['delivery\_date']=detail['delivery\_date']

details1['invoice\_date']=info['invoice\_date']

details1['p\_inv\_num']=info['p\_inv\_num']

tax\_get=Sup\_Upload1.objects.get(upload\_name=details1['productname'])

details1['tax']=float(tax\_get.upload\_tax)

details1['total\_amt\_wtax']=format((float(amt)+(float(amt))\*(float(details1['tax']/100))),'.2f')

est\_qty\_list.append(details1)

dealer\_name=DealerOrderData.objects.get(po\_num=info['po\_num'])

for prod in dealer\_name.product\_info:

company\_data=Company\_Details.objects().only('company\_name','company\_address','mobile','email','gstin','state','pan','cin')

company\_info=json.loads(company\_data.to\_json())

for i in company\_info:

info['company\_details']=i

dealer\_info=Reg\_Dealer.objects.get(id=dealer\_name.userid)

info['po\_num']=dealer\_name.po\_num

info['rfq\_num']=dealer\_name.rfq\_num

info['invoice\_status']=info['invoice\_status']

info['po\_gen\_date']=str(dealer\_name.po\_gen\_date)

info['qty']=prod.qty

info['dealer\_name']=dealer\_info.dealer\_name

info['dealer\_city']=dealer\_info.dealer\_city

info['sm\_name']=dealer\_name.sales\_manager

info['sm\_mobile']=dealer\_name.sales\_manager\_mobile

info['moreinv\_info']=est\_qty\_list

totalprice=sum(float(prod['total\_amt']) for prod in est\_qty\_list)

total\_price=format(totalprice,'.2f')

return{"Status":"Success","more\_Info":est\_qty\_info,"total\_price":total\_price}

else:

return {"status":"No records found"}

if usertype=='accountant':

chalan\_check= Invoice\_Info.objects(invoice\_status='Final Inv Pending').count()

print chalan\_check,'chalan\_check'

if chalan\_check >0:

est\_qty= Invoice\_Info.objects(invoice\_status='Final Inv Pending').only('po\_num','p\_inv\_num','invoice\_date','invoice\_status','moreinv\_info\_\_avl\_carton','moreinv\_info\_\_piece\_per\_carton','moreinv\_info\_\_total\_amt','moreinv\_info\_\_productname','moreinv\_info\_\_req\_product\_qty','moreinv\_info\_\_invoice\_status','moreinv\_info\_\_remain\_qty','delivery\_type','moreinv\_info\_\_split\_inv\_info\_\_delivery\_date','moreinv\_info\_\_split\_inv\_info\_\_unique\_num','moreinv\_info\_\_split\_inv\_info\_\_acpt\_inv\_qty','moreinv\_info\_\_split\_inv\_info\_\_p\_inv\_qty','chalan\_num','driver\_name','dl\_num','driver\_contact','vehicle\_num','courier\_num','courier\_name','courier\_boy','weight','moreinv\_info\_\_split\_inv\_info\_\_inv\_status','moreinv\_info\_\_split\_inv\_info\_\_restrictions','moreinv\_info\_\_split\_inv\_info\_\_check\_qty')

est\_qty\_info=json.loads(est\_qty.to\_json())

for info in est\_qty\_info:

est\_qty\_list=[]

for data in info['moreinv\_info']:

for detail in data['split\_inv\_info']:

details1={}

details1['p\_inv\_qty']=detail['p\_inv\_qty']

if int(detail['p\_inv\_qty'])>0:

details1['total\_amt']=format(float(data['total\_amt']),'.2f')

amt=data['total\_amt']

details1['req\_product\_qty']=data['req\_product\_qty']

details1['remain\_qty']=data['remain\_qty']

details1['pieceperCorton']=data['piece\_per\_carton']

details1['productname']=data['productname']

details1['unique\_num']=detail['unique\_num']

details1["estimated\_pieceperCarton"]=int(data['req\_product\_qty'])/int(data['piece\_per\_carton'])

details1['avl\_qty']=data['avl\_carton']

details1['check\_qty']=detail['check\_qty']

details1['acpt\_inv\_qty']=detail['acpt\_inv\_qty']

details1['delivery\_date']=detail['delivery\_date']

details1['invoice\_date']=info['invoice\_date']

details1['p\_inv\_num']=info['p\_inv\_num']

tax\_get=Sup\_Upload1.objects.get(upload\_name=details1['productname'])

details1['tax']=float(tax\_get.upload\_tax)

details1['total\_amt\_wtax']=format((float(amt)+(float(amt))\*(float(details1['tax']/100))),'.2f')

est\_qty\_list.append(details1)

dealer\_name=DealerOrderData.objects.get(po\_num=info['po\_num'])

for prod in dealer\_name.product\_info:

company\_data=Company\_Details.objects().only('company\_name','company\_address','mobile','email','gstin','state','pan','cin')

company\_info=json.loads(company\_data.to\_json())

for i in company\_info:

info['company\_details']=i

dealer\_info=Reg\_Dealer.objects.get(id=dealer\_name.userid)

info['po\_num']=dealer\_name.po\_num

info['rfq\_num']=dealer\_name.rfq\_num

info['invoice\_status']=info['invoice\_status']

info['po\_gen\_date']=str(dealer\_name.po\_gen\_date)

info['qty']=prod.qty

for bill in dealer\_info.billing\_address:

bill\_addr={}

bill\_addr['d\_no']=bill.d\_no

bill\_addr['street']=bill.street

bill\_addr['city']=bill.city

bill\_addr['state']=bill.state

bill\_addr['zipcode']=bill.zipcode

ship\_list=[]

for ship in dealer\_info.shipping\_address:

ship\_addr={}

ship\_addr['d\_no']=ship.d\_no

ship\_addr['street']=ship.street

ship\_addr['city']=ship.city

ship\_addr['state']=ship.state

ship\_addr['zipcode']=ship.zipcode

ship\_list.append(ship\_addr)

info['billing\_address']=bill\_addr

info['shipping\_address']=ship\_list

info['dealer\_name']=dealer\_info.dealer\_name

info['dealer\_city']=dealer\_info.dealer\_city

info['sm\_name']=dealer\_name.sales\_manager

info['sm\_mobile']=dealer\_name.sales\_manager\_mobile

info['moreinv\_info']=est\_qty\_list

totalprice=sum(float(prod['total\_amt']) for prod in est\_qty\_list)

total\_price=format(totalprice,'.2f')

return{"Status":"Success","more\_Info":est\_qty\_info,"total\_price":total\_price}

else:

return {"status":"No records found"}

else:

return {"status":"Success","user\_check":"invalid usertype"}

else:

return {"status":"invalid secret key"}

def put(self):

json\_data=request.get\_json(force=True)

po\_num=json\_data['po\_num']

for items in json\_data['estimated\_qty\_info']:

prod\_name=items['product\_name']

inv\_no=items['inv\_num']

est\_qty=items['avl\_carton']

unic\_num=items['unique\_num']

restrictions=items['restrictions']if 'restrictions' in json\_data else ''

check\_qty=items['check\_qty']

po\_check=Invoice\_Info.objects.get(po\_num=po\_num,p\_inv\_num=inv\_no)

for more in po\_check.moreinv\_info:

for inv in more.split\_inv\_info:

if more.productname==prod\_name and unic\_num==inv.unique\_num:

more.avl\_carton=str(est\_qty)

inv.check\_qty=str(check\_qty)

inv.restrictions=str(restrictions)

more.save()

if more.productname==prod\_name and inv.unique\_num=='0':

more.avl\_carton=str(est\_qty)

inv.check\_qty=str(check\_qty)

inv.restrictions=str(restrictions)

more.save()

po\_check.invoice\_status='Final Inv Pending'

po\_check.save()

return {"status":"qty updated successfully."}

class InvPreparation(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

rfq\_num=request.args['rfq\_num']

#info\_list=[]

rfq\_object=DealerOrderData.objects.get(status="Approved",rfq\_num=rfq\_num)

rfq\_list=[]

more\_info=[]

for data in rfq\_object.product\_info:

userid=rfq\_object.userid

dealer\_info=Reg\_Dealer.objects.get(id=userid)

details1={}

details1["dealer\_name"]=dealer\_info.dealer\_name

details1["dealer\_city"]=dealer\_info.dealer\_city

details1["dealer\_mail"]=dealer\_info.dealer\_mail

details1["dealer\_mobile"]=dealer\_info.dealer\_mobile

details1['rfq\_generate\_date']=(rfq\_object.rfq\_generate\_date).strftime("%d-%m-%Y")

details1["rfq\_num"]=rfq\_object.rfq\_num

details1["productid"]=data.productid

details1["productname"]=data.productname

details1["modelno"]=data.modelno

details1["brand"]=data.brand

details1["qty"]=data.qty

details1["mrp"]=data.mrp

details1["total"]=data.total

details1["discount"]=data.discount

details1["volume\_disc"]=data.volume\_disc

details1["savings"]=data.savings

details1["spcl\_disc"]=data.spcl\_disc

if data.status=='Pending':

details1["product\_status"]='Pending for PO'

split\_list=[]

for k in data.split\_products:

details1["accept\_productname"]=k.accept\_productname

details1["balance\_qty"]=k.balance\_qty

details1["split\_status"]=k.split\_status

details1["product\_status"]=k.product\_status

details1["dealer\_status"]=k.dealer\_status

details1["rfq\_num"]=rfq\_object.rfq\_num

if data.productname==k.accept\_productname and k.product\_status=='Split':

info={"remarks\_dealer":k.remarks\_dealer,"remarks\_internal":k.remarks\_internal,"split\_delvry\_date":k.split\_delvry\_date,"unique\_nbr":k.unique\_nbr,"accept\_prod\_qty":k.accept\_prod\_qty}

split\_info=json.loads(json.dumps(info))

split\_list.append(split\_info)

details1["split\_info"]=split\_list

more\_info.append(details1)

details1["rfq\_lists"]=more\_info

info=json.loads(json.dumps(details1))

rfq\_list.append(details1)

return {"status":"Success","info\_list":rfq\_list}

class Rfq\_History(Resource):

def get(self):

#if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

user\_id=request.args['userid']

more\_info=[]

for rfq\_objects in DealerOrderData.objects(userid=user\_id,status\_\_ne="Pending"):

#more\_info=[]

for data in rfq\_objects.product\_info:

details={}

details["productid"]=data.productid

details["productname"]=data.productname

details["modelno"]=data.modelno

details["brand"]=data.brand

details["qty"]=data.qty

details["mrp"]=data.mrp

details["total"]=data.total

details["discount"]=data.discount

details["rfq\_num"]=rfq\_objects.rfq\_num

details["volume\_disc"]=data.volume\_disc

details["savings"]=data.savings

details["spcl\_disc"]=data.spcl\_disc

if data.status=='Pending':

details["product\_status"]='Pending for PO'

split\_list=[]

for k in data.split\_products:

details["accept\_productname"]=k.accept\_productname

details["balance\_qty"]=k.balance\_qty

details["split\_status"]=k.split\_status

details["product\_status"]=k.product\_status

details["dealer\_status"]=k.dealer\_status

details["unique\_nbr"]=k.unique\_nbr

details["rfq\_num"]=rfq\_objects.rfq\_num

if data.productname==k.accept\_productname and k.product\_status=='Split':

info={"remarks\_dealer":k.remarks\_dealer,"remarks\_internal":k.remarks\_internal,"rfq\_num":rfq\_objects.rfq\_num,"split\_delvry\_date":k.split\_delvry\_date,"unique\_nbr":k.unique\_nbr,"accept\_prod\_qty":k.accept\_prod\_qty}

split\_list.append(info)

details["split\_info"]=split\_list

more\_info.append(details)

return {"status":"success","more\_info":more\_info}

class Proforma\_Inv\_Print(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

po\_num=request.args['po\_num']

usertype=request.args['usertype']

if usertype=='accountant' or usertype=='whManager' or usertype=='salesManager' or usertype=='Dealer':

inv\_check= Invoice\_Info.objects(po\_num=po\_num).count()

if inv\_check >0:

rfq={}

invoice= Invoice\_Info.objects(po\_num=po\_num).only('po\_num','p\_inv\_num','invoice\_status','moreinv\_info\_\_split\_inv\_info\_\_restrictions','moreinv\_info\_\_piece\_per\_carton','moreinv\_info\_\_avl\_carton','moreinv\_info\_\_total\_amt','moreinv\_info\_\_productname','moreinv\_info\_\_req\_product\_qty','moreinv\_info\_\_invoice\_status','moreinv\_info\_\_remain\_qty','moreinv\_info\_\_split\_inv\_info\_\_p\_inv\_qty')

chalan\_info=json.loads(invoice.to\_json())

for info in chalan\_info:

moreinv\_info\_list=[]

total=0.0

for data in info['moreinv\_info']:

for detail in data['split\_inv\_info']:

details1={}

details1['req\_product\_qty']=data['req\_product\_qty']

details1['remain\_qty']=data['remain\_qty']

if int(detail['p\_inv\_qty'])>0:

if data['avl\_carton']=='':

details1['acpt\_carton']=''

if data['avl\_carton']!='':

details1['acpt\_carton']=data['avl\_carton']

details1['p\_inv\_qty']=detail['p\_inv\_qty']

details1['restrictions']=detail['restrictions']

details1['productname']=data['productname']

details1['est\_qty']=int(data['req\_product\_qty'])/int(data['piece\_per\_carton'])

tax=Sup\_Upload1.objects.get(upload\_name=details1['productname'])

details1['tax']=tax.upload\_tax

dealer\_name=DealerOrderData.objects.get(po\_num=info['po\_num'])

for prod in dealer\_name.product\_info:

if prod.productname==data['productname']:

details1['mrp']=prod.mrp

details1['volume\_disc']=prod.volume\_disc

details1['spcl\_disc']=prod.spcl\_disc

print details1['spcl\_disc']

#details1['discount']=float(100)-float(prod.savings)

#details1['product\_amt']=float(details1['mrp'])-(float(details1['mrp'])\*(float(details1['discount']))/100)

details1['product\_amt']=float(data['total\_amt'])

details1['discount']=tax.upload\_discount

details1['brand']=tax.upload\_brand

details1['total\_amt']=format(float(details1['product\_amt']+(float(details1['product\_amt'])\*(float(details1['tax'])/100))),'.2f')

moreinv\_info\_list.append(details1)

dealer\_name=DealerOrderData.objects.get(po\_num=info['po\_num'])

for prod in dealer\_name.product\_info:

company\_data=Company\_Details.objects().only('company\_name','company\_address','mobile','email','gstin','state','pan','cin')

company\_info=json.loads(company\_data.to\_json())

for i in company\_info:

dealer={}

dealer['company\_details']=i

totalprice=sum(float(prod['total\_amt']) for prod in moreinv\_info\_list)

total\_price=format(totalprice,'.2f')

info['grand\_total']=total\_price

dealer\_info=Reg\_Dealer.objects.get(id=dealer\_name.userid)

rfq['rfq\_num']=dealer\_name.rfq\_num

rfq['order\_date']=(dealer\_name.rfq\_generate\_date).strftime("%d/%m/%Y")

rfq['invoice\_status']=info['invoice\_status']

rfq['po\_num']=info['po\_num']

dealer['dealer\_mobile']=dealer\_info.dealer\_mobile

dealer['shop\_name']=dealer\_info.dealer\_mobile

for bill in dealer\_info.billing\_address:

bill\_addr={}

bill\_addr['d\_no']=bill.d\_no

bill\_addr['street']=bill.street

bill\_addr['city']=bill.city

bill\_addr['state']=bill.state

bill\_addr['zipcode']=bill.zipcode

ship\_list=[]

for ship in dealer\_info.shipping\_address:

ship\_addr={}

ship\_addr['d\_no']=ship.d\_no

ship\_addr['street']=ship.street

ship\_addr['city']=ship.city

ship\_addr['state']=ship.state

ship\_addr['zipcode']=ship.zipcode

#ship\_list.append(ship\_addr)

dealer['billing\_address']=bill\_addr

dealer['shipping\_address']=ship\_addr

dealer['dealer\_name']=dealer\_info.dealer\_name

dealer['dealer\_city']=dealer\_info.dealer\_city

info['moreinv\_info']=moreinv\_info\_list

return{"Status":"Success","more\_Info":chalan\_info,"dealer":[dealer],"rfq":rfq}

else:

return {"status":"No records found"}

else:

return {"status":"Success","user\_check":"invalid usertype"}

else:

return {"status":"invalid secret key"}

class Company\_Details\_Saving(Resource):

def post(self):

saving\_data=Company\_Details(company\_name='LN TOOLS PRIVATE LIMITED',company\_address=['7-3-112', 'RVENUE', 'GAGAN PAHAD','KATEDAN-HYD','RANGA REDDY DISTRICT',' TELANGANA-500077'],mobile='9290571112',email='lntoolsprivateltd@gmail.com',gstin='36AADCL3443J1ZH',state='36 Telangana',pan='AADCL3443J',cin='U74999TG2017PTC117829').save()

return {"status":"data saved successfully"}

def get(self):

company\_data=Company\_Details.objects().only('company\_name','company\_address','mobile','email','gstin','state','pan','cin')

info=json.loads(company\_data.to\_json())

for data in info:

for i in data['company\_address']:

dic={}

dic['asdf']=i

return {"status":"success","company\_info":info}

class ChalanGeneration(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

usertype=request.args['usertype']

#po\_num=request.args['po\_num']

if usertype=='whManager' or usertype=='accountant' or usertype=='Dealer':

chalan\_check= Invoice\_Info.objects(invoice\_status='Final Inv Generated',chalan\_status\_\_ne='Ch-Generated').count()

print chalan\_check,'chalan\_check'

if chalan\_check >0:

billing\_address={}

shipping\_address={}

est\_qty= Invoice\_Info.objects(invoice\_status='Final Inv Generated',chalan\_status\_\_ne='Ch-Generated').only('po\_num','chalan\_status','f\_inv\_num','p\_inv\_num','invoice\_date','invoice\_status','moreinv\_info\_\_avl\_carton','moreinv\_info\_\_piece\_per\_carton','moreinv\_info\_\_total\_amt','moreinv\_info\_\_productname','moreinv\_info\_\_req\_product\_qty','moreinv\_info\_\_invoice\_status','moreinv\_info\_\_remain\_qty','delivery\_type','moreinv\_info\_\_split\_inv\_info\_\_delivery\_date','moreinv\_info\_\_split\_inv\_info\_\_unique\_num','moreinv\_info\_\_split\_inv\_info\_\_acpt\_inv\_qty','moreinv\_info\_\_split\_inv\_info\_\_p\_inv\_qty','moreinv\_info\_\_split\_inv\_info\_\_f\_inv\_num','chalan\_num','driver\_name','dl\_num','driver\_contact','vehicle\_num','courier\_num','courier\_name','courier\_boy','weight','moreinv\_info\_\_split\_inv\_info\_\_inv\_status','moreinv\_info\_\_split\_inv\_info\_\_acpt\_carton')

est\_qty\_info=json.loads(est\_qty.to\_json())

for info in est\_qty\_info:

est\_qty\_list=[]

for data in info['moreinv\_info']:

for detail in data['split\_inv\_info']:

details1={}

details1['p\_inv\_qty']=detail['p\_inv\_qty']

details1['chalan\_num']=info['chalan\_num']

if int(details1['p\_inv\_qty'])>0 and info['chalan\_num']=="":

details1['po\_num']=info['po\_num']

details1['total\_amt']=data['total\_amt']

details1['req\_product\_qty']=data['req\_product\_qty']

details1['remain\_qty']=data['remain\_qty']

details1['p\_inv\_num']=info['p\_inv\_num']

details1['f\_inv\_num']=info['f\_inv\_num']

details1['acpt\_carton']=detail['acpt\_carton']

details1['pieceperCorton']=data['piece\_per\_carton']

details1['productname']=data['productname']

details1["estimated\_pieceperCarton"]=int(data['req\_product\_qty'])/int(data['piece\_per\_carton'])

details1['avl\_qty']=data['avl\_carton']

details1['acpt\_inv\_qty']=detail['acpt\_inv\_qty']

details1['unique\_num']=detail['unique\_num']

details1['delivery\_date']=detail['delivery\_date']

details1['delivery\_type']=info['delivery\_type']

details1['driver\_name']=info['driver\_name']

details1['dl\_num']=info['dl\_num']

print data['productname']

#prod\_data=Sup\_Upload1.objects.get(upload\_name=data['productname'])

amount=format(float(data['avl\_carton'])\*float(data['piece\_per\_carton'])\*float(details1['total\_amt']),'.2f')

details1['final\_amount']=amount

details1['driver\_contact']=info['driver\_contact']

details1['vehicle\_num']=info['vehicle\_num']

details1['courier\_num']=info['courier\_num']

details1['courier\_name']=info['courier\_name']

details1['courier\_boy']=info['courier\_boy']

details1['weight']=info['weight']

details1['chalan\_status']=info['chalan\_status']

details1['invoice\_date']=info['invoice\_date']

est\_qty\_list.append(details1)

dealer\_name=DealerOrderData.objects.get(po\_num=info['po\_num'])

for prod in dealer\_name.product\_info:

company\_data=Company\_Details.objects().only('company\_name','company\_address','mobile','email','gstin','state','pan','cin')

company\_info=json.loads(company\_data.to\_json())

for i in company\_info:

info['company\_details']=i

dealer\_info=Reg\_Dealer.objects.get(id=dealer\_name.userid)

info['po\_num']=dealer\_name.po\_num

info['rfq\_num']=dealer\_name.rfq\_num

info['invoice\_status']=info['invoice\_status']

info['po\_gen\_date']=str(dealer\_name.po\_gen\_date)

info['qty']=prod.qty

for bill in dealer\_info.billing\_address:

bill\_addr={}

bill\_addr['d\_no']=bill.d\_no

bill\_addr['street']=bill.street

bill\_addr['city']=bill.city

bill\_addr['state']=bill.state

bill\_addr['zipcode']=bill.zipcode

ship\_list=[]

for ship in dealer\_info.shipping\_address:

ship\_addr={}

ship\_addr['d\_no']=ship.d\_no

ship\_addr['street']=ship.street

ship\_addr['city']=ship.city

ship\_addr['state']=ship.state

ship\_addr['zipcode']=ship.zipcode

ship\_list.append(ship\_addr)

billing\_address['billing\_address']=bill\_addr

shipping\_address['shipping\_address']=ship\_addr

info['dealer\_name']=dealer\_info.dealer\_name

info['dealer\_city']=dealer\_info.dealer\_city

info['sm\_name']=dealer\_name.sales\_manager

info['sm\_mobile']=dealer\_name.sales\_manager\_mobile

info['moreinv\_info']=est\_qty\_list

totalprice=sum(float(prod['total\_amt']) for prod in est\_qty\_list)

total\_price=format(totalprice,'.2f')

return{"Status":"Success","more\_Info":est\_qty\_info,"total\_price":total\_price,"shipping\_address":shipping\_address,"billing\_address":billing\_address}

else:

return {"status":"No records found"}

else:

return {"status":"Success","user\_check":"invalid usertype"}

else:

return {"status":"invalid secret key"}

class Inv\_Data(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args=request.args

if args['usertype']=='accountant':

chalan\_check= Invoice\_Info.objects(Q(invoice\_status='Final Inv Generated') | Q(invoice\_status='Final Inv Pending')).count()

print chalan\_check,'chalan\_check'

if chalan\_check >0:

est\_qty= Invoice\_Info.objects(Q(invoice\_status='Final Inv Generated') | Q(invoice\_status='Final Inv Pending')).only('po\_num','invoice\_date','p\_inv\_num','invoice\_status','moreinv\_info\_\_avl\_carton','moreinv\_info\_\_piece\_per\_carton','moreinv\_info\_\_total\_amt','moreinv\_info\_\_productname','moreinv\_info\_\_req\_product\_qty','moreinv\_info\_\_invoice\_status','moreinv\_info\_\_remain\_qty','delivery\_type','moreinv\_info\_\_split\_inv\_info\_\_delivery\_date','moreinv\_info\_\_split\_inv\_info\_\_unique\_num','moreinv\_info\_\_split\_inv\_info\_\_acpt\_inv\_qty','moreinv\_info\_\_split\_inv\_info\_\_p\_inv\_qty','chalan\_num','driver\_name','dl\_num','driver\_contact','vehicle\_num','courier\_num','courier\_name','courier\_boy','weight','moreinv\_info\_\_split\_inv\_info\_\_inv\_status')

est\_qty\_info=json.loads(est\_qty.to\_json())

est\_qty\_list=[]

for info in est\_qty\_info:

for data in info['moreinv\_info']:

for detail in data['split\_inv\_info']:

details1={}

details1['p\_inv\_qty']=detail['p\_inv\_qty']

if int(details1['p\_inv\_qty'])>0:

details1['total\_amt']=data['total\_amt']

details1['req\_product\_qty']=data['req\_product\_qty']

details1['remain\_qty']=data['remain\_qty']

details1['p\_inv\_num']=info['p\_inv\_num']

details1['pieceperCorton']=data['piece\_per\_carton']

details1['productname']=data['productname']

details1["estimated\_pieceperCarton"]=int(data['req\_product\_qty'])/int(data['piece\_per\_carton'])

details1['avl\_qty']=data['avl\_carton']

details1['acpt\_inv\_qty']=detail['acpt\_inv\_qty']

details1['delivery\_date']=detail['delivery\_date']

details1['invoice\_date']=info['invoice\_date']

est\_qty\_list.append(details1)

dealer\_name=DealerOrderData.objects.get(po\_num=info['po\_num'])

for prod in dealer\_name.product\_info:

company\_data=Company\_Details.objects().only('company\_name','company\_address','mobile','email','gstin','state','pan','cin')

company\_info=json.loads(company\_data.to\_json())

for i in company\_info:

info['company\_details']=i

dealer\_info=Reg\_Dealer.objects.get(id=dealer\_name.userid)

info['po\_num']=dealer\_name.po\_num

info['rfq\_num']=dealer\_name.rfq\_num

info['invoice\_status']=info['invoice\_status']

info['po\_gen\_date']=str(dealer\_name.po\_gen\_date)

info['qty']=prod.qty

for bill in dealer\_info.billing\_address:

bill\_addr={}

bill\_addr['d\_no']=bill.d\_no

bill\_addr['street']=bill.street

bill\_addr['city']=bill.city

bill\_addr['state']=bill.state

bill\_addr['zipcode']=bill.zipcode

ship\_list=[]

for ship in dealer\_info.shipping\_address:

ship\_addr={}

ship\_addr['d\_no']=ship.d\_no

ship\_addr['street']=ship.street

ship\_addr['city']=ship.city

ship\_addr['state']=ship.state

ship\_addr['zipcode']=ship.zipcode

ship\_list.append(ship\_addr)

info['billing\_address']=bill\_addr

info['shipping\_address']=ship\_list

info['dealer\_name']=dealer\_info.dealer\_name

info['dealer\_city']=dealer\_info.dealer\_city

info['sm\_name']=dealer\_name.sales\_manager

info['sm\_mobile']=dealer\_name.sales\_manager\_mobile

info['moreinv\_info']=est\_qty\_list

totalprice=sum(float(prod['total\_amt']) for prod in est\_qty\_list)

total\_price=format(totalprice,'.2f')

return{"Status":"Success","more\_Info":est\_qty\_info,"total\_price":total\_price}

else:

return {"status":"No records found"}

else:

return {"status":"invalid secret key"}

class UserProfile(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

userid=request.args['userid']

usertype=request.args['usertype']

if usertype=='Dealer':

dealer\_data=Reg\_Dealer.objects.get(id=userid)

details={}

details['dealer\_name']=dealer\_data.dealer\_name

details['dealer\_mobile']=dealer\_data.dealer\_mobile

details['dealer\_mail']=dealer\_data.dealer\_mail

details['default\_address']=dealer\_data.default\_address

for data in dealer\_data.billing\_address:

bill={'dno':data.d\_no,'street':data.street,'city':data.city,'state':data.state,'zipcode':data.zipcode}

details['billing\_address']=bill

ship\_addr=[]

for ship\_data in dealer\_data.shipping\_address:

ship={'dno':ship\_data.d\_no,'street':ship\_data.street,'city':ship\_data.city,'state':ship\_data.state,'zipcode':ship\_data.zipcode}

ship\_addr.append(ship)

details['shipping\_address']=ship\_addr

details['reg\_shopname']=dealer\_data.reg\_shopname

return {"status":"Success","profile\_info":details}

if usertype=='web':

users=CustomerDetails.objects.get(id=userid)

details={}

details['firstname']=users.firstname

details['lastname']=users.lastname

details['gst\_number']=users.gst\_number

details['email']=users.email

details['mobile']=users.mobile

bill\_addr=[]

for data in users.billing\_address:

bill={'full\_name':data.full\_name,'mobile':data.mobile,'dno':data.d\_no,'street':data.street,'city':data.city,'state':data.state,'zipcode':data.zipcode,"addunique\_nbr":data.addunique\_nbr}

bill\_addr.append(bill)

details['billing\_address']=bill\_addr

print details['billing\_address']

ship\_addr=[]

for ship\_data in users.shipping\_address:

ship={'full\_name':ship\_data.full\_name,'mobile':ship\_data.mobile,'dno':ship\_data.d\_no,'street':ship\_data.street,'city':ship\_data.city,'state':ship\_data.state,'zipcode':ship\_data.zipcode,"addunique\_nbr":ship\_data.addunique\_nbr}

ship\_addr.append(ship)

details['shipping\_address']=ship\_addr

return {"status":"Success","profile\_info":details}

else:

return {"status":"Invalid secret key"}

def put(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args = request.args

json\_data = request.get\_json(force=True)

shipping\_address =json\_data['shipping\_address']

billing\_address =json\_data['billing\_address']

usertype =json\_data['usertype']

if usertype=='Dealer':

user\_info = Reg\_Dealer.objects.get(id=args['userid'])

if user\_info:

if len(user\_info.shipping\_address)==0:

for acpt\_prod in json\_data['shipping\_address']:

d\_no=acpt\_prod['d\_no']

street=acpt\_prod['street']

city=acpt\_prod['city']

state=acpt\_prod['state']

zipcode=acpt\_prod['zipcode']

datadb=ShippingAddress1(d\_no,street,city,state,zipcode)

user\_info.shipping\_address.append(datadb)

user\_info.save()

return {"status":"details added successfully"}

if len(user\_info.shipping\_address)>0:

for data in user\_info.shipping\_address:

for acpt\_prod in json\_data['shipping\_address']:

check=any(acpt\_prod['d\_no'] in x['d\_no'] for x in user\_info.shipping\_address)#to chk dicatiory data present in list

if check==True:

data.d\_no=acpt\_prod['d\_no']

data.street=acpt\_prod['street']

data.city=acpt\_prod['city']

data.state=acpt\_prod['state']

data.zipcode=acpt\_prod['zipcode']

data.save()

return {"status":"details added successfully"}

if check==False:

d\_no=acpt\_prod['d\_no']

street=acpt\_prod['street']

city=acpt\_prod['city']

state=acpt\_prod['state']

zipcode=acpt\_prod['zipcode']

datadb=ShippingAddress1(d\_no,street,city,state,zipcode)

user\_info.shipping\_address.append(datadb)

user\_info.save()

return {"status":"details added successfully"}

if usertype=='web':

customer\_info = CustomerDetails.objects.get(id=args['userid'])

if customer\_info:

if len(customer\_info.shipping\_address)==0:

for acpt\_prod in json\_data['shipping\_address']:

full\_name=acpt\_prod['full\_name']

mobile=acpt\_prod['mobile']

d\_no=acpt\_prod['d\_no']

street=acpt\_prod['street']

city=acpt\_prod['city']

state=acpt\_prod['state']

zipcode=acpt\_prod['zipcode']

addunique\_nbr=acpt\_prod['addunique\_nbr']

listofno = random.sample(set([0,1,2,3,4,5,6,7,8,9]),10)

randomnumber="".join(str(numbers) for numbers in listofno)

addunique\_nbr=randomnumber

datadb=CustShippingAddress(full\_name,mobile,d\_no,street,city,state,zipcode,addunique\_nbr)

customer\_info.shipping\_address.append(datadb)

customer\_info.save()

if len(customer\_info.billing\_address)==0:

for acpt\_prod in json\_data['billing\_address']:

full\_name=acpt\_prod['full\_name']

print full\_name

mobile=acpt\_prod['mobile']

print mobile

d\_no=acpt\_prod['d\_no']

street=acpt\_prod['street']

city=acpt\_prod['city']

state=acpt\_prod['state']

zipcode=acpt\_prod['zipcode']

addunique\_nbr=acpt\_prod['addunique\_nbr']

listofno = random.sample(set([0,1,2,3,4,5,6,7,8,9]),10)

randomnumber="".join(str(numbers) for numbers in listofno)

addunique\_nbr=randomnumber

datadb=CustBillingAddress(full\_name,mobile,d\_no,street,city,state,zipcode,addunique\_nbr)

customer\_info.billing\_address.append(datadb)

customer\_info.save()

return {"status":"details added successfully"}

if len(customer\_info.shipping\_address)>0 or len(customer\_info.billing\_address)>0:

for data in customer\_info.shipping\_address:

for acpt\_prod in json\_data['shipping\_address']:

check=any(acpt\_prod['addunique\_nbr'] in x['addunique\_nbr'] for x in customer\_info.shipping\_address)#to chk dicatiory data present in list

if check==True:

print 'shiptrue'

data.full\_name=acpt\_prod['full\_name']

data.mobile=acpt\_prod['mobile']

data.d\_no=acpt\_prod['d\_no']

data.street=acpt\_prod['street']

data.city=acpt\_prod['city']

data.state=acpt\_prod['state']

data.zipcode=acpt\_prod['zipcode']

data.save()

for data in customer\_info.billing\_address:

for acpt\_prod in json\_data['billing\_address']:

checkbill=any(acpt\_prod['addunique\_nbr'] in x['addunique\_nbr'] for x in customer\_info.billing\_address)#to chk dicatiory data present in list

if checkbill==True:

print 'billtrue'

data.full\_name=acpt\_prod['full\_name']

data.mobile=acpt\_prod['mobile']

data.d\_no=acpt\_prod['d\_no']

data.street=acpt\_prod['street']

data.city=acpt\_prod['city']

data.state=acpt\_prod['state']

data.zipcode=acpt\_prod['zipcode']

data.save()

return {"status":"details added successfully"}

else:

return {"status":"Invalid secret key"}

def delete(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

ship\_address=json\_data['shipping\_address']

bill\_address=json\_data['billing\_address']

userid=json\_data['userid']

usertype =json\_data['usertype']

if usertype=='Dealer':

dealer\_ship\_data=Reg\_Dealer.objects(id=userid)

if dealer\_ship\_data.count()>0:

dealer\_info=Reg\_Dealer.objects.get(id=userid)

for data in ship\_address:

for i in dealer\_info.shipping\_address:

if data['d\_no']==i['d\_no']:

dealer\_info.shipping\_address.remove(i)

dealer\_info.save()

return{"status":"Deleted Successfully"}

if usertype=='web':

customer\_ship\_info = CustomerDetails.objects(id=userid)

if customer\_ship\_info.count()>0:

customer\_info = CustomerDetails.objects.get(id=userid)

for data in json\_data['shipping\_address']:

for i in customer\_info.shipping\_address:

if data['d\_no']==i['d\_no']:

customer\_info.shipping\_address.remove(i)

customer\_info.save()

for datas in json\_data['billing\_address']:

for j in customer\_info.billing\_address:

if datas['d\_no']==j['d\_no']:

customer\_info.billing\_address.remove(i)

customer\_info.save()

return{"status":"Deleted Successfully"}

else:

return {"status":"Invalid secret key"}

from collections import defaultdict

import pandas as pd

class Print\_Invoice\_Info(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

inv\_num=request.args['p\_inv\_num']

usertype=request.args['usertype']

if usertype=='accountant':

inv\_check= Invoice\_Info.objects(p\_inv\_num=inv\_num).count()

if inv\_check >0:

rfq={}

snowfall = defaultdict(float)

invoice= Invoice\_Info.objects(p\_inv\_num=inv\_num).only('f\_inv\_num','po\_num','p\_inv\_num','invoice\_date','invoice\_status','moreinv\_info\_\_split\_inv\_info\_\_restrictions','moreinv\_info\_\_piece\_per\_carton','moreinv\_info\_\_avl\_carton','moreinv\_info\_\_total\_amt','moreinv\_info\_\_productname','moreinv\_info\_\_req\_product\_qty','moreinv\_info\_\_invoice\_status','moreinv\_info\_\_remain\_qty','moreinv\_info\_\_split\_inv\_info\_\_p\_inv\_qty')

chalan\_info=json.loads(invoice.to\_json())

for info in chalan\_info:

rfq['invoice\_date']=info['invoice\_date']

rfq['p\_inv\_num']=info['p\_inv\_num']

rfq['f\_inv\_num']=info['f\_inv\_num']

moreinv\_info\_list=[]

list1=[]

total=0.0

for data in info['moreinv\_info']:

for detail in data['split\_inv\_info']:

dic1={}

details1={}

details1['req\_product\_qty']=data['req\_product\_qty']

details1['remain\_qty']=data['remain\_qty']

if int(detail['p\_inv\_qty'])>0:

if data['avl\_carton']=='':

details1['acpt\_carton']=''

if data['avl\_carton']!='':

details1['acpt\_carton']=data['avl\_carton']

details1['p\_inv\_qty']=detail['p\_inv\_qty']

details1['restrictions']=detail['restrictions']

details1['productname']=data['productname']

dic1['productname']=data['productname']

dic1['p\_inv\_qty']=detail['p\_inv\_qty']

details1['est\_qty']=int(data['req\_product\_qty'])/int(data['piece\_per\_carton'])

tax=Sup\_Upload1.objects.get(upload\_name=details1['productname'])

details1['tax']=tax.upload\_tax

dealer\_name=DealerOrderData.objects.get(po\_num=info['po\_num'])

for prod in dealer\_name.product\_info:

if prod.productname==data['productname']:

details1['mrp']=prod.mrp

details1['spcl\_disc']=prod.spcl\_disc

details1['volume\_disc']=prod.volume\_disc

#details1['discount']=float(100)-float(prod.savings)

#details1['product\_amt']=float(details1['mrp'])-(float(details1['mrp'])\*(float(details1['discount']))/100)

details1['product\_amt']=format(float(data['total\_amt']),'.2f')

dic1['product\_amt']=float(data['total\_amt'])

amt=float(data['total\_amt'])

details1['discount']=tax.upload\_discount

details1['brand']=tax.upload\_brand

details1['total\_amt']=format(float(amt+(float(amt)\*(float(details1['tax'])/100))),'.2f')

moreinv\_info\_list.append(details1)

list1.append(dic1)

#dictf = reduce(lambda x, y: dict((k, v + y[k]) for k, v in x.iteritems()), list1)

for i in list1:

snowfall[i['productname']] += i['product\_amt']

snowfall = [{'productname': year, 'total': snowfall[year], 'qty': snowfall[year]}

for year in sorted(snowfall, reverse=True)]

#print snowfall

#information=pd.DataFrame.from\_records(list1).astype(int).groupby('productname').sum().T.to\_dict()

#print information

dealer\_name=DealerOrderData.objects.get(po\_num=info['po\_num'])

for prod in dealer\_name.product\_info:

company\_data=Company\_Details.objects().only('company\_name','company\_address','mobile','email','gstin','state','pan','cin')

company\_info=json.loads(company\_data.to\_json())

for i in company\_info:

dealer={}

dealer['company\_details']=i

totalprice=sum(float(prod['total\_amt']) for prod in moreinv\_info\_list)

total\_price=format(totalprice,'.2f')

info['grand\_total']=total\_price

totalamt=sum(float(prod['product\_amt']) for prod in moreinv\_info\_list)

total\_amt=format(totalamt,'.2f')

info['total\_sum']=float(total\_amt)

info['tax\_amt']=format((float(info['total\_sum'])\*(float(details1['tax'])/100)),'.2f')

info["tax\_amt\_grand"]=float(info['total\_sum'])+float(info['tax\_amt'])

p = inflect.engine()

val= int(round(info["tax\_amt\_grand"],0))

#p.number\_to\_words(val)

info["tax\_amt\_words"]=p.number\_to\_words(val).replace(',','')

print info["tax\_amt\_words"]

dealer\_info=Reg\_Dealer.objects.get(id=dealer\_name.userid)

rfq['rfq\_num']=dealer\_name.rfq\_num

rfq['order\_date']=(dealer\_name.rfq\_generate\_date).strftime("%d/%m/%Y")

rfq['invoice\_status']=info['invoice\_status']

rfq['po\_num']=info['po\_num']

dealer['dealer\_mobile']=dealer\_info.dealer\_mobile

dealer['shop\_name']=dealer\_info.dealer\_shopname

for bill in dealer\_info.billing\_address:

bill\_addr={}

bill\_addr['d\_no']=bill.d\_no

bill\_addr['street']=bill.street

bill\_addr['city']=bill.city

bill\_addr['state']=bill.state

bill\_addr['zipcode']=bill.zipcode

ship\_list=[]

for ship in dealer\_info.shipping\_address:

ship\_addr={}

ship\_addr['d\_no']=ship.d\_no

ship\_addr['street']=ship.street

ship\_addr['city']=ship.city

ship\_addr['state']=ship.state

ship\_addr['zipcode']=ship.zipcode

#ship\_list.append(ship\_addr)

dealer['billing\_address']=bill\_addr

dealer['shipping\_address']=ship\_addr

dealer['dealer\_name']=dealer\_info.dealer\_name

dealer['dealer\_city']=dealer\_info.dealer\_city

info['moreinv\_info']=moreinv\_info\_list

return{"Status":"Success","more\_Info":chalan\_info,"dealer":[dealer],"rfq":rfq,"snowfall":snowfall}

else:

return {"status":"No records found"}

else:

return {"status":"Success","user\_check":"invalid usertype"}

else:

return {"status":"invalid secret key"}

class Chalan\_Details(Resource):

def get(self):

chalan\_data=Invoice\_Info.objects(chalan\_status='Ch-Generated').only('remarks','po\_num','chalan\_status','f\_inv\_num','p\_inv\_num','invoice\_date','invoice\_status','moreinv\_info\_\_avl\_carton','moreinv\_info\_\_piece\_per\_carton','moreinv\_info\_\_total\_amt','moreinv\_info\_\_productname','moreinv\_info\_\_req\_product\_qty','moreinv\_info\_\_invoice\_status','moreinv\_info\_\_remain\_qty','delivery\_type','moreinv\_info\_\_split\_inv\_info\_\_delivery\_date','moreinv\_info\_\_split\_inv\_info\_\_unique\_num','moreinv\_info\_\_split\_inv\_info\_\_acpt\_inv\_qty','moreinv\_info\_\_split\_inv\_info\_\_p\_inv\_qty','moreinv\_info\_\_split\_inv\_info\_\_f\_inv\_num','chalan\_num','driver\_name','dl\_num','driver\_contact','vehicle\_num','courier\_num','courier\_name','courier\_boy','weight','moreinv\_info\_\_split\_inv\_info\_\_inv\_status','moreinv\_info\_\_split\_inv\_info\_\_acpt\_carton')

chalan\_list=json.loads(chalan\_data.to\_json())

if chalan\_data:

for i in chalan\_list:

dealer\_name=DealerOrderData.objects.get(po\_num=i['po\_num'])

dealer\_info=Reg\_Dealer.objects.get(id=dealer\_name.userid)

i['dealer\_name']=dealer\_info.dealer\_name

i['dealer\_mobile']=dealer\_info.dealer\_mobile

i['dealer\_city']=dealer\_info.dealer\_city

for bill in dealer\_info.billing\_address:

bill\_addr={}

bill\_addr['d\_no']=bill.d\_no

bill\_addr['street']=bill.street

bill\_addr['city']=bill.city

bill\_addr['state']=bill.state

bill\_addr['zipcode']=bill.zipcode

ship\_list=[]

for ship in dealer\_info.shipping\_address:

ship\_addr={}

ship\_addr['d\_no']=ship.d\_no

ship\_addr['street']=ship.street

ship\_addr['city']=ship.city

ship\_addr['state']=ship.state

ship\_addr['zipcode']=ship.zipcode

i['billing\_address']=bill\_addr

i['shipping\_address']=ship\_addr

return {"status":"success","chalan\_list":chalan\_list}

else:

return {"status":"chalan doesnot exists"}

class DealerRegisterHistory(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

dealers\_list=[]

for dealer\_objects in Reg\_Dealer.objects():

dealers\_data=json.loads(dealer\_objects.to\_json())

dealers\_list.append(dealers\_data)

return{"status":"success","Dealer\_history":dealers\_list}

class UserProfile(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

userid=request.args['userid']

usertype=request.args['usertype']

if usertype=='Dealer':

dealer\_data=Reg\_Dealer.objects.get(id=userid)

details={}

details['dealer\_name']=dealer\_data.dealer\_name

details['dealer\_mobile']=dealer\_data.dealer\_mobile

details['dealer\_mail']=dealer\_data.dealer\_mail

details['default\_address']=dealer\_data.default\_address

for data in dealer\_data.billing\_address:

bill={'dno':data.d\_no,'street':data.street,'city':data.city,'state':data.state,'zipcode':data.zipcode}

details['billing\_address']=bill

ship\_addr=[]

for ship\_data in dealer\_data.shipping\_address:

ship={'dno':ship\_data.d\_no,'street':ship\_data.street,'city':ship\_data.city,'state':ship\_data.state,'zipcode':ship\_data.zipcode}

ship\_addr.append(ship)

details['shipping\_address']=ship\_addr

details['reg\_shopname']=dealer\_data.reg\_shopname

return {"status":"Success","profile\_info":details}

if usertype=='web':

users=CustomerDetails.objects.get(id=userid)

details={}

details['firstname']=users.firstname

details['lastname']=users.lastname

details['gst\_number']=users.gst\_number

details['email']=users.email

details['mobile']=users.mobile

bill\_addr=[]

for data in users.billing\_address:

bill={'full\_name':data.full\_name,'mobile':data.mobile,'dno':data.d\_no,'street':data.street,'city':data.city,'state':data.state,'zipcode':data.zipcode,"addunique\_nbr":data.addunique\_nbr}

bill\_addr.append(bill)

details['billing\_address']=bill\_addr

print details['billing\_address']

ship\_addr=[]

for ship\_data in users.shipping\_address:

ship={'full\_name':ship\_data.full\_name,'mobile':ship\_data.mobile,'dno':ship\_data.d\_no,'street':ship\_data.street,'city':ship\_data.city,'state':ship\_data.state,'zipcode':ship\_data.zipcode,"addunique\_nbr":ship\_data.addunique\_nbr}

ship\_addr.append(ship)

details['shipping\_address']=ship\_addr

return {"status":"Success","profile\_info":details}

else:

return {"status":"Invalid secret key"}

def put(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

args = request.args

json\_data = request.get\_json(force=True)

shipping\_address =json\_data['shipping\_address']

billing\_address =json\_data['billing\_address']

usertype =json\_data['usertype']

if usertype=='Dealer':

user\_info = Reg\_Dealer.objects.get(id=args['userid'])

if user\_info:

if len(user\_info.shipping\_address)==0:

for acpt\_prod in json\_data['shipping\_address']:

d\_no=acpt\_prod['d\_no']

street=acpt\_prod['street']

city=acpt\_prod['city']

state=acpt\_prod['state']

zipcode=acpt\_prod['zipcode']

datadb=ShippingAddress1(d\_no,street,city,state,zipcode)

user\_info.shipping\_address.append(datadb)

user\_info.save()

return {"status":"details added successfully"}

if len(user\_info.shipping\_address)>0:

for data in user\_info.shipping\_address:

for acpt\_prod in json\_data['shipping\_address']:

check=any(acpt\_prod['d\_no'] in x['d\_no'] for x in user\_info.shipping\_address)#to chk dicatiory data present in list

if check==True:

data.d\_no=acpt\_prod['d\_no']

data.street=acpt\_prod['street']

data.city=acpt\_prod['city']

data.state=acpt\_prod['state']

data.zipcode=acpt\_prod['zipcode']

data.save()

return {"status":"details added successfully"}

if check==False:

d\_no=acpt\_prod['d\_no']

street=acpt\_prod['street']

city=acpt\_prod['city']

state=acpt\_prod['state']

zipcode=acpt\_prod['zipcode']

datadb=ShippingAddress1(d\_no,street,city,state,zipcode)

user\_info.shipping\_address.append(datadb)

user\_info.save()

return {"status":"details added successfully"}

if usertype=='web':

customer\_info = CustomerDetails.objects.get(id=args['userid'])

if customer\_info:

if len(customer\_info.shipping\_address)==0:

for acpt\_prod in json\_data['shipping\_address']:

full\_name=acpt\_prod['full\_name']

mobile=acpt\_prod['mobile']

d\_no=acpt\_prod['d\_no']

street=acpt\_prod['street']

city=acpt\_prod['city']

state=acpt\_prod['state']

zipcode=acpt\_prod['zipcode']

addunique\_nbr=acpt\_prod['addunique\_nbr']

listofno = random.sample(set([0,1,2,3,4,5,6,7,8,9]),10)

randomnumber="".join(str(numbers) for numbers in listofno)

addunique\_nbr=randomnumber

datadb=CustShippingAddress(full\_name,mobile,d\_no,street,city,state,zipcode,addunique\_nbr)

customer\_info.shipping\_address.append(datadb)

customer\_info.save()

if len(customer\_info.billing\_address)==0:

for acpt\_prod in json\_data['billing\_address']:

full\_name=acpt\_prod['full\_name']

print full\_name

mobile=acpt\_prod['mobile']

print mobile

d\_no=acpt\_prod['d\_no']

street=acpt\_prod['street']

city=acpt\_prod['city']

state=acpt\_prod['state']

zipcode=acpt\_prod['zipcode']

addunique\_nbr=acpt\_prod['addunique\_nbr']

listofno = random.sample(set([0,1,2,3,4,5,6,7,8,9]),10)

randomnumber="".join(str(numbers) for numbers in listofno)

addunique\_nbr=randomnumber

datadb=CustBillingAddress(full\_name,mobile,d\_no,street,city,state,zipcode,addunique\_nbr)

customer\_info.billing\_address.append(datadb)

customer\_info.save()

return {"status":"details added successfully"}

if len(customer\_info.shipping\_address)>0 or len(customer\_info.billing\_address)>0:

for data in customer\_info.shipping\_address:

for acpt\_prod in json\_data['shipping\_address']:

check=any(acpt\_prod['addunique\_nbr'] in x['addunique\_nbr'] for x in customer\_info.shipping\_address)#to chk dicatiory data present in list

if check==True:

print 'shiptrue'

data.full\_name=acpt\_prod['full\_name']

data.mobile=acpt\_prod['mobile']

data.d\_no=acpt\_prod['d\_no']

data.street=acpt\_prod['street']

data.city=acpt\_prod['city']

data.state=acpt\_prod['state']

data.zipcode=acpt\_prod['zipcode']

data.save()

for data in customer\_info.billing\_address:

for acpt\_prod in json\_data['billing\_address']:

checkbill=any(acpt\_prod['addunique\_nbr'] in x['addunique\_nbr'] for x in customer\_info.billing\_address)#to chk dicatiory data present in list

if checkbill==True:

print 'billtrue'

data.full\_name=acpt\_prod['full\_name']

data.mobile=acpt\_prod['mobile']

data.d\_no=acpt\_prod['d\_no']

data.street=acpt\_prod['street']

data.city=acpt\_prod['city']

data.state=acpt\_prod['state']

data.zipcode=acpt\_prod['zipcode']

data.save()

return {"status":"details added successfully"}

else:

return {"status":"Invalid secret key"}

def delete(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data=request.get\_json(force=True)

ship\_address=json\_data['shipping\_address']

bill\_address=json\_data['billing\_address']

userid=json\_data['userid']

usertype =json\_data['usertype']

if usertype=='Dealer':

dealer\_ship\_data=Reg\_Dealer.objects(id=userid)

if dealer\_ship\_data.count()>0:

dealer\_info=Reg\_Dealer.objects.get(id=userid)

for data in ship\_address:

for i in dealer\_info.shipping\_address:

if data['d\_no']==i['d\_no']:

dealer\_info.shipping\_address.remove(i)

dealer\_info.save()

return{"status":"Deleted Successfully"}

if usertype=='web':

customer\_ship\_info = CustomerDetails.objects(id=userid)

if customer\_ship\_info.count()>0:

customer\_info = CustomerDetails.objects.get(id=userid)

for data in json\_data['shipping\_address']:

for i in customer\_info.shipping\_address:

if data['d\_no']==i['d\_no']:

customer\_info.shipping\_address.remove(i)

customer\_info.save()

for datas in json\_data['billing\_address']:

for j in customer\_info.billing\_address:

if datas['d\_no']==j['d\_no']:

customer\_info.billing\_address.remove(i)

customer\_info.save()

return{"status":"Deleted Successfully"}

else:

return {"status":"Invalid secret key"}

class DealerRegisterHistory(Resource):

def get(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

dealers\_list=[]

for dealer\_objects in Reg\_Dealer.objects():

dealers\_data=json.loads(dealer\_objects.to\_json())

dealers\_list.append(dealers\_data)

return{"status":"success","Dealer\_history":dealers\_list}

class DealerVerifyOTP(Resource):

def put(self):

pool = Pool()

json\_data = request.get\_json(force=True)

userid=json\_data['userid']

mobile=json\_data['mobile']

usertype=json\_data['usertype']

otp=json\_data['otp']

print otp

if usertype=='Dealer':

check\_otp=DealerMobileUpdate.objects.get(dealer\_mobile=mobile,dealer\_otp=otp)

if check\_otp:

user\_info = Reg\_Dealer.objects.get(id=userid)

if user\_info>0:

user\_info.dealer\_mobile=mobile

user\_info.save()

check\_otp.delete()

return {"status":"Mobile Number Updated successfully."}

if usertype=='web':

cust\_otp=CustomerMobileUpdate.objects.get(cust\_mobile=mobile,cust\_otp=otp)

if cust\_otp:

cust\_info=CustomerDetails.objects.get(id=userid)

if cust\_info>0:

cust\_info.cust\_mobile=mobile

cust\_info.save()

cust\_otp.delete()

return {"status":"Mobile Number Updated successfully."}

def post(self):

if request.headers.has\_key('secret\_key') and request.headers['secret\_key']==secret\_key:

json\_data = request.get\_json(force=True)

userid=json\_data['userid']

mobile=json\_data['mobile']

usertype=json\_data['usertype']

if usertype=='Dealer':

user\_info = Reg\_Dealer.objects.get(id=userid)

my\_secret = 'MFRGGZDFMZTWQ2LK'

otp\_is = otp.get\_totp(my\_secret)

if user\_info:

user\_info.dealer\_mobile=mobile

msg='Your one time password is '+str(otp\_is)

mobile=mobile

print mobile

#pool.apply\_async(sms,[msg,mobile])

sms(msg,mobile)

DealerMobileUpdate(userid=str(user\_info.id),dealer\_mobile=mobile,dealer\_otp=str(otp\_is)).save()

return {"status":"Mobile Number Saved successfully.","otp":str(otp\_is),"mobile":mobile}

if usertype=='web':

cust\_info = CustomerDetails.objects.get(id=userid)

my\_secret = 'MFRGGZDFMZTWQ2LK'

otp\_is = otp.get\_totp(my\_secret)

if cust\_info:

cust\_info.mobile=mobile

msg='Your one time password is '+str(otp\_is)

mobile=mobile

print mobile

#pool.apply\_async(sms,[msg,mobile])

sms(msg,mobile)

DealerMobileUpdate(userid=str(user\_info.id),dealer\_mobile=mobile,dealer\_otp=str(otp\_is)).save()

return {"status":"Mobile Number Saved successfully.","otp":str(otp\_is),"mobile":mobile}

class AddressDefault(Resource):

def put(self):

json\_data = request.get\_json(force=True)

userid=json\_data['userid']

user\_type=json\_data['user\_type']

d\_no=json\_data['d\_no']

addunique\_nbr=json\_data['addunique\_nbr']

if user\_type=='Dealer':

dealer\_info=Reg\_Dealer.objects.get(id=userid)

for i,j in enumerate(dealer\_info.shipping\_address):

if d\_no==j['d\_no']:

dealer\_info.default\_address=str(i)

dealer\_info.save()

return {"Status":"Address Set as Default"}

if user\_type=='web':

customer\_info = CustomerDetails.objects.get(id=userid)

for i,j in enumerate(customer\_info.shipping\_address):

if addunique\_nbr==j['addunique\_nbr']:

customer\_info.ship\_default\_address=str(i)

customer\_info.save()

for k,l in enumerate(customer\_info.billing\_address):

if addunique\_nbr==l['addunique\_nbr']:

customer\_info.bill\_default\_address=str(k)

customer\_info.save()

return {"Status":"Address Set as Default"}