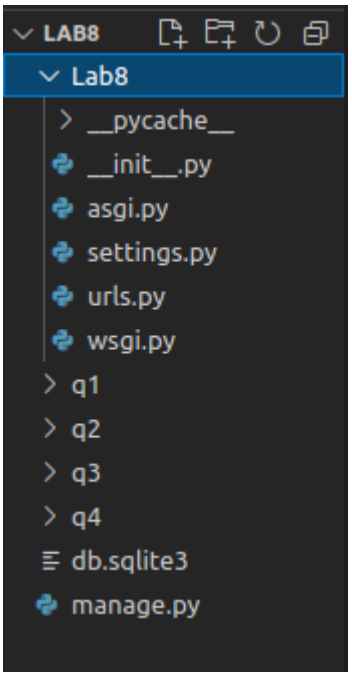
SANSKAAR PATNI
180905134 CSE C 23
IT LAB 8 REST API

Created a Project Lab8 and made 4 apps each app representing one question q1,q2,q3,q4
Folder Structure



Modified parts in Lab8/Lab8 folder

settings.py

```
INSTALLED_APPS = [
    'q1',
    'q2',
    'q3',
    'q4',
    'rest_framework',
    'django.contrib.admin',
```

urls.py

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('q1/', include('q1.urls')),
    path('q2/', include('q2.urls')),
    path('q3/', include('q3.urls')),
```

```
    path('q4/', include('q4.urls')),
]
```

1.
Lab8/q1/
models.py

```
from django.db import models


class User(models.Model):
    identity = models.CharField(max_length=20, default="ANONYMOUS")
    registered_on = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.identity


class Blog(models.Model):
    blog_text = models.CharField(max_length=1000)
    blog_created_by = models.ForeignKey(
        User, related_name='userBlog', on_delete=models.CASCADE)
    pub_date = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.blog_text


class Comment(models.Model):
    blog = models.ForeignKey(Blog, related_name='blog',
                             on_delete=models.CASCADE)
    comment_text = models.CharField(max_length=1000)
    write_date = models.DateTimeField(auto_now=True)

    def __str__(self):
        return self.comment_text
```

serializers.py

```
from rest_framework import serializers
from .models import Blog, Comment, User


class BlogSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Blog
```

```
        fields = ('blog_text', 'blog_created_by', 'pub_date')


class CommentSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Comment
        fields = ('comment_text', 'blog', 'write_date')


class UserSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = User
        fields = ('id', 'identity', 'registered_on')
```

views.py

```
from rest_framework import viewsets
from .serializers import UserSerializer, CommentSerializer,
BlogSerializer
from .models import User, Comment, Blog
from rest_framework.decorators import action
from rest_framework.response import Response
from rest_framework import status

#showing the respected HTTP methods as asked in question
#that too are automatically done by viewset in rest framework
class blogs_list(viewsets.ModelViewSet):
    queryset = Blog.objects.all()
    serializer_class = BlogSerializer

    @action(detail=True, methods=['put'])
    def blog_update(self, request, pk):
        try:
            blogObj = Blog.objects.get(pk=pk)
        except:
            return Response(status=status.HTTP_404_NOT_FOND)
        serializer = BlogSerializer(blogObj, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.error_messages,
status=status.HTTP_400_BAD_REQUEST)
```

```
class users_list(viewsets.ModelViewSet):
   queryset = User.objects.all()
   serializer_class = UserSerializer

   @action(detail=True, methods=['post'])
   def user_registration(self, request):
       serializer = UserSerializer(data=request.data)
       if serializer.is_valid():
           serializer.save()
           return Response(serializer.data,
status=status.HTTP_201_CREATED)
       return Response(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)


class comments_list(viewsets.ModelViewSet):
   queryset = Comment.objects.all()
   serializer_class = CommentSerializer

   @action(detail=True, methods=['post'])
   def comment_add(self, request):
       serializer = CommentSerializer(data=request.data)
       if serializer.is_valid():
           serializer.save()
           return Response(serializer.data,
status=status.HTTP_201_CREATED)
       return Response(serializer.errors,
status=status.HTTP_400_BAD_REQUEST)

   @action(detail=True, methods=['delete'])
   def comment_delete(self, request, pk):
       try:
           commentObj = Comment.objects.get(pk=pk)
       except:
           return Response(status=status.HTTP_404_NOT_FOUND)

       commentObj.delete()
       return Response(status=status.HTTP_204_NO_CONTENT)
```

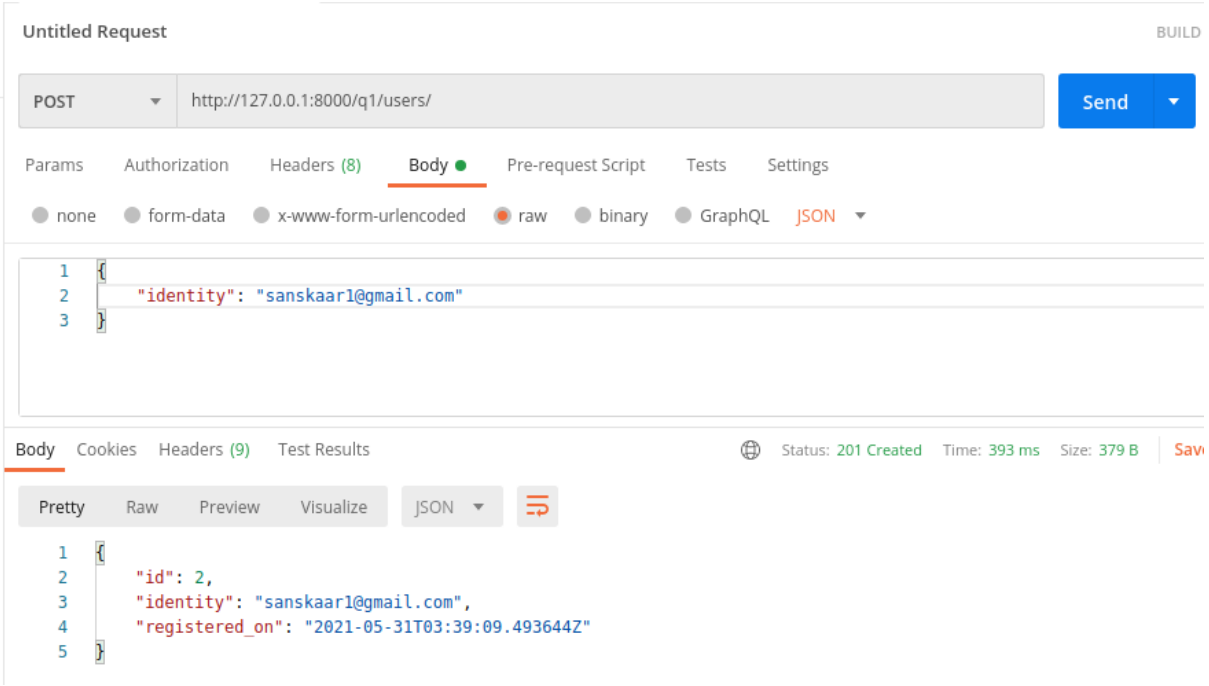
urls.py

```
from django.urls import path, include
```

```
from . import views
from rest_framework import routers

router = routers.DefaultRouter()
router.register(r'blogs', views.blogs_list)
router.register(r'users', views.users_list)
router.register(r'comments', views.comments_list)

urlpatterns = [
    path('', include(router.urls)),
    path('api-auth/', include('rest_framework.urls',
namespace='rest_framework')),
]
```

Asked Screenshots in the question:

User registration



Update existing blog.

Untitled Request

BUILD

GET http://127.0.0.1:8000/q1/blogs/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "identity": "sanskaar1@gmail.com"
3 }
```

Body Cookies Headers (9) Test Results Status: 200 OK Time: 8 ms Size: 422 B Save Re

Pretty Raw Preview Visualize JSON

```
1 [
2     {
3         "blog_text": "Sanskaar's First Blog",
4         "blog_created_by": "http://127.0.0.1:8000/q1/users/1/",
5         "pub_date": "2021-05-31T03:36:41.622534Z"
6     }
7 ]
```

PUT http://127.0.0.1:8000/q1/blogs/1/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "blog_text": "Sanskaar's First Blog(Updated)",
3     "blog_created_by": "http://127.0.0.1:8000/q1/users/1/"
4 }
```

Body Cookies Headers (9) Test Results Status: 200 OK Time: 358 ms Size: 443 B Save

Pretty Raw Preview Visualize JSON

```
1 {
2     "blog_text": "Sanskaar's First Blog(Updated)",
3     "blog_created_by": "http://127.0.0.1:8000/q1/users/1/",
4     "pub_date": "2021-05-31T03:40:54.398027Z"
5 }
```

Add a comment.

POST http://127.0.0.1:8000/q1/comments/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "comment_text": "Concise information packed first blog, Sanskaar!",
    "blog": "http://127.0.0.1:8000/q1/blogs/1/"
}
```

Body Cookies Headers (9) Test Results Status: 201 Created Time: 330 ms Size: 446 B Sav

Pretty Raw Preview Visualize JSON

```
{
    "comment_text": "Concise information packed first blog, Sanskaar!",
    "blog": "http://127.0.0.1:8000/q1/blogs/1/",
    "write_date": "2021-05-31T03:44:09.129444Z"
}
```

Delete comment

GET http://127.0.0.1:8000/q1/comments/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

Body Cookies Headers (9) Test Results Status: 200 OK Time: 6 ms Size: 588 B Sav

Pretty Raw Preview Visualize JSON

```
[
    {
        "comment_text": "Very well written first blog, Sanskaar!",
        "blog": "http://127.0.0.1:8000/q1/blogs/1/",
        "write_date": "2021-05-31T03:37:23.225399Z"
    },
    {
        "comment_text": "Concise information packed first blog, Sanskaar!",
        "blog": "http://127.0.0.1:8000/q1/blogs/1/",
        "write_date": "2021-05-31T03:44:09.129444Z"
    }
]
```

2.
Lab8/q2/
models.py

```
from django.db import models
from django.urls.conf import path
from django.utils.functional import keep_lazy


class Customer(models.Model):
    name = models.CharField(max_length=50)
    longitude = models.FloatField()
    latitude = models.FloatField()


class Cab(models.Model):
    model = models.CharField(max_length=100)
    noOfpass = models.IntegerField()
    baseFare = models.IntegerField()
    farePerKmAfterBaseDistance = models.IntegerField()
    longitude = models.FloatField()
    latitude = models.FloatField()
```

serializers.py

```
from rest_framework import serializers
from .models import Customer, Cab


class CustomerSerializer(serializers.ModelSerializer):
    class Meta:
        model = Customer
```

```
        fields = ('id', 'longitude', 'latitude')


class CabSerializer(serializers.ModelSerializer):
    class Meta:
        model = Cab
        fields = ('id', 'model', 'noOfpass', 'baseFare',
                  'farePerKmAfterBaseDistance', 'longitude', 'latitude')
```

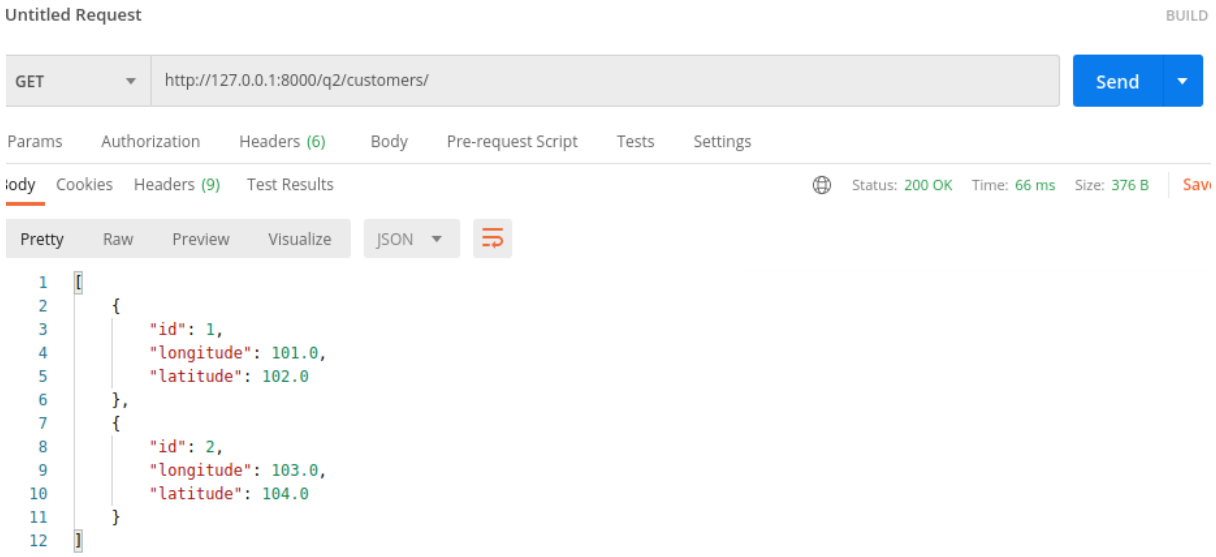
views.py

```
from rest_framework import viewsets
from .serializers import CustomerSerializer, CabSerializer
import math
from .models import Customer, Cab
from rest_framework.decorators import api_view
from rest_framework.response import Response


class customers_list(viewsets.ModelViewSet):
    queryset = Customer.objects.all()
    serializer_class = CustomerSerializer


class cabs_list(viewsets.ModelViewSet):
    queryset = Cab.objects.all()
    serializer_class = CabSerializer

# Cab Speed = 2
@api_view(['GET'])
def get_eta(request, pk):
    person = Customer.objects.get(pk=pk)
    cabs = Cab.objects.all()
    fare_dict = []
    for cab in cabs:
        xdist = person.latitude - cab.latitude
        ydist = person.longitude - cab.longitude
        dist = math.sqrt(xdist*xdist+ydist*ydist)
        eta = dist/2
        fare_det = {'Distance': dist, "ETA": eta}
        fare_dict.append(fare_det)
    return Response(fare_dict)
```

urls.py

```python
from django.urls import path, include
from . import views
from rest_framework import routers

router = routers.DefaultRouter()
router.register(r'customers', views.customers_list)
router.register(r'cabs', views.cabs_list)

urlpatterns = [
    path('', include(router.urls)),
    path('eta/<int:pk>/', views.get_eta, name='get_eta'),
    path('api-auth/', include('rest_framework.urls',
namespace='rest_framework')),
]
```

Screenshots:(Already used POST to insert data)
GET for customers:



GET for cabs:

Untitled Request

BUILD

GET http://127.0.0.1:8000/q2/cabs/ Send Save

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies Co

Body Cookies Headers (9) Test Results Status: 200 OK Time: 116 ms Size: 526 B Save Response

Pretty Raw Preview Visualize JSON


```
2   {
3       "id": 1,
4       "model": "Alto",
5       "noOfpass": 2,
6       "baseFare": 20,
7       "farePerKmAfterBaseDistance": 10,
8       "longitude": 102.0,
9       "latitude": 103.0
10  },
11  {
12      "id": 2,
13      "model": "Amaze",
14      "noOfpass": 3,
15      "baseFare": 30,
16      "farePerKmAfterBaseDistance": 13,
17      "longitude": 104.0,
18      "latitude": 103.0
19  }
```


Estimated Time of Arrival (ETA) for customer 1:

Untitled Request

BUILD

GET http://127.0.0.1:8000/q2/eta/1 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (9) Test Results Status: 200 OK Time: 7 ms Size: 391 B Save

Pretty Raw Preview Visualize JSON


```
1   [
2       {
3           "Distance": 1.4142135623730951,
4           "ETA": 0.7071067811865476
5       },
6       {
7           "Distance": 3.1622776601683795,
8           "ETA": 1.5811388300841898
9       }
10  ]
```


Fare Details for a particular cab:

Untitled Request

BUILD

GET http://127.0.0.1:8000/q2/cabs/1 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (9) Test Results Status: 200 OK Time: 8 ms Size: 419 B Save

Pretty Raw Preview Visualize JSON


```
{
    "id": 1,
    "model": "Alto",
    "noOfpass": 2,
    "baseFare": 20,
    "farePerKmAfterBaseDistance": 10,
    "longitude": 102.0,
    "latitude": 103.0
}
```


3.
Lab8/q3/
models.py

```
from django.db import models
from django.core.validators import MinValueValidator, MaxValueValidator


class Dish(models.Model):
    name = models.CharField(max_length=50)
    description = models.CharField(max_length=1000)
    price = models.IntegerField()

    def __str__(self):
        return self.name


class Restaurant(models.Model):
    name = models.CharField(max_length=1000)
    avgrating = models.IntegerField(validators=[MinValueValidator(0),
                                                MaxValueValidator(5)])
    description = models.CharField(max_length=1000)
    menu = models.ManyToManyField(Dish)
    address = models.CharField(max_length=100)
    cuisine = models.CharField(max_length=50, default='Italian')

    def __str__(self):
        return self.name


class Review(models.Model):
```

```
    review_by = models.CharField(max_length=100)
    text = models.CharField(max_length=1000)
    restaurant = models.ForeignKey(
        Restaurant, on_delete=models.CASCADE)

    def __str__(self):
        return self.text
```

serializers.py

```
from rest_framework import serializers
from .models import Restaurant, Dish, Review


class DishSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Dish
        fields = '__all__'


class RestaurantSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Restaurant
        fields = '__all__'


class ReviewSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Review
        fields = '__all__'
```

views.py

```
from rest_framework import viewsets
from .serializers import DishSerializer, RestaurantSerializer,
ReviewSerializer
from .models import Restaurant, Dish, Review
from django.http.response import JsonResponse
from rest_framework.decorators import api_view

# viewset of rest framework takes care of crud operations 3 resources
dish,restaurants, reviews


class dish(viewsets.ModelViewSet):
```

```
    queryset = Dish.objects.all()
    serializer_class = DishSerializer


class restaurant(viewsets.ModelViewSet):
    queryset = Restaurant.objects.all()
    serializer_class = RestaurantSerializer


class review(viewsets.ModelViewSet):
    queryset = Review.objects.all()
    serializer_class = ReviewSerializer

# functions for searching a restaurant by location, name, cuisine and
also dish name

@api_view(['GET'])
def getRestaurantByLocation(request, location):
    restaurants =
Restaurant.objects.filter(address__startswith=location)
    serializer = RestaurantSerializer(
        restaurants, context={'request': request}, many=True)
    return JsonResponse(serializer.data, safe=False)


@api_view(['GET'])
def getRestaurantByName(request, name):
    restaurants = Restaurant.objects.filter(name__startswith=name)
    serializer = RestaurantSerializer(
        restaurants, context={'request': request}, many=True)
    return JsonResponse(serializer.data, safe=False)


@api_view(['GET'])
def getRestaurantByCuisine(request, cuisine):
    restaurants = Restaurant.objects.filter(cuisine__startswith=cuisine)
    serializer = RestaurantSerializer(
        restaurants, context={'request': request}, many=True)
    return JsonResponse(serializer.data, safe=False)


@api_view(['GET'])
def getRestaurantByDishName(request, dish):
```

```
    restaurants = Restaurant.objects.filter(menu__name__startswith=dish)
    serializer = RestaurantSerializer(
        restaurants, context={'request': request}, many=True)
    return JsonResponse(serializer.data, safe=False)
```

urls.py

```
from django.urls import path, include
from . import views
from rest_framework import routers

router = routers.DefaultRouter()

router.register(r'dishes', views.dish)
router.register(r'restaurants', views.restaurant)
router.register(r'reviews', views.review)

urlpatterns = [
    path('', include(router.urls)),
    path('search/location/<str:location>',
         views.getRestaurantByLocation, name="getRestaurantByLocation"),
    path('search/name/<str:name>', views.getRestaurantByName,
         name="getRestaurantByName"),
    path('search/cuisine/<str:cuisine>', views.getRestaurantByCuisine,
         name="getRestaurantByCuisine"),
    path('search/cuisine/<str:dish>', views.getRestaurantByDishName,
         name="getRestaurantByDishName"),
]
```

Screenshots:
CRUD operations can be applied to restaurants, dishes, reviews:

Restaurant CRUD operation:
GET:

Untitled Request

BUILD

GET http://127.0.0.1:8000/q3/list/restaurants/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Body Cookies Headers (9) Test Results Status: 200 OK Time: 12 ms Size: 1.16 KB Sav

Pretty Raw Preview Visualize JSON


```
[
    {
        "url": "http://127.0.0.1:8000/q3/list/restaurants/1/",
        "name": "McDonald's",
        "avgrating": 4,
        "description": "Fast Food Restaurant",
        "address": "Andheri, Mumbai",
        "cuisine": "American",
        "menu": [
            "http://127.0.0.1:8000/q3/list/dishes/1/",
            "http://127.0.0.1:8000/q3/list/dishes/2/",
            "http://127.0.0.1:8000/q3/list/dishes/3/"
        ]
    },
    {
        "url": "http://127.0.0.1:8000/q3/list/restaurants/2/",
        "name": "Pizza Hut",
        "avgrating": 4,
        "description": "Pizza Hut is an American restaurant chain and international franchise founded in 1958.",
        "address": "Andheri, Mumbai",
        "cuisine": "Italian",
        "menu": [
            "http://127.0.0.1:8000/q3/list/dishes/2/"
        ]
    },
    {
        "url": "http://127.0.0.1:8000/q3/list/restaurants/3/",
        "name": "Dominoes Pizza",
        "avgrating": 4,
        "description": "Domino's is the largest pizza restaurant chain in the world based on global retail sales.",
        "address": "Dadar, Mumbai",
```


POST(Screenshot of Dominoes being added):

Untitled Request

BUILD

POST http://127.0.0.1:8000/q3/list/restaurants/ Send Save

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Bea


```
{
    "name": "Dominoes Pizza",
    "avgrating": 4,
    "description": "Domino's is the largest pizza restaurant chain in the world based on global retail sales.",
    "address": "Dadar, Mumbai",
    "cuisine": "Italian",
    "menu": [
        "http://127.0.0.1:8000/q3/list/dishes/2/"
    ]
```


Body Cookies Headers (9) Test Results Status: 200 OK Time: 12 ms Size: 1.16 KB Save Respon

Pretty Raw Preview Visualize JSON


```
        "avgrating": 4,
        "description": "Domino's is the largest pizza restaurant chain in the world based on global retail sales.",
        "address": "Dadar, Mumbai",
        "cuisine": "Italian",
        "menu": [
            "http://127.0.0.1:8000/q3/list/dishes/2/"
```


PUT(Deleting Margherita Pizza from McDonald's):

Untitled Request

BUILD

PUT http://127.0.0.1:8000/q3/list/restaurants/1/ Send Save

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beau

Body Cookies Headers (9) Test Results Status: 200 OK Time: 483 ms Size: 569 B Save Respons

Pretty Raw Preview Visualize JSON


```
{
    "url": "http://127.0.0.1:8000/q3/list/restaurants/1/",
    "name": "McDonald's",
    "avgrating": 4,
    "description": "Fast Food Restaurant",
    "address": "Andheri, Mumbai",
    "cuisine": "American",
    "menu": [
        "http://127.0.0.1:8000/q3/list/dishes/1/",
        "http://127.0.0.1:8000/q3/list/dishes/3/"
    ]
}
```


DELETE(Pizza Hut):

Untitled Request

BUILD

DELETE http://127.0.0.1:8000/q3/list/restaurants/2 Send Save

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (9) Test Results Status: 200 OK Time: 28 ms Size: 591 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "url": "http://127.0.0.1:8000/q3/list/restaurants/2/",
    "name": "Pizza Hut",
```


After deleting restaurant list:

Untitled Request

BUILD

GET http://127.0.0.1:8000/q3/list/restaurants/ Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (9) Test Results Status: 200 OK Time: 13 ms

Pretty Raw Preview Visualize JSON


```
[
    {
        "url": "http://127.0.0.1:8000/q3/list/restaurants/1/",
        "name": "McDonald's",
        "avgrating": 4,
        "description": "Fast Food Restaurant",
        "address": "Andheri, Mumbai",
        "cuisine": "American",
        "menu": [
            "http://127.0.0.1:8000/q3/list/dishes/1/",
            "http://127.0.0.1:8000/q3/list/dishes/3/"
        ]
    },
    {
        "url": "http://127.0.0.1:8000/q3/list/restaurants/3/",
        "name": "Dominoes Pizza",
        "avgrating": 4,
        "description": "Domino's is the largest pizza restaurant chain in the world based on global retail sales.",
        "address": "Dadar, Mumbai",
        "cuisine": "Italian",
        "menu": [
            "http://127.0.0.1:8000/q3/list/dishes/2/"
        ]
    }
]
```


Similar for CRUD for dishes,reviews at their respective end points

Search by name:

Untitled Request

BUILD

GET http://127.0.0.1:8000/q3/search/name/Mc Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Body Cookies Headers (9) Test Results Status: 200 OK Time: 19 ms Size: 559 B Save

Pretty Raw Preview Visualize JSON


```
[
    {
        "url": "http://127.0.0.1:8000/q3/list/restaurants/1/",
        "name": "McDonald's",
        "avgrating": 4,
        "description": "Fast Food Restaurant",
        "address": "Andheri, Mumbai",
        "cuisine": "American",
        "menu": [
            "http://127.0.0.1:8000/q3/list/dishes/1/",
            "http://127.0.0.1:8000/q3/list/dishes/3/"
        ]
    }
]
```


Search by location:

Untitled Request

BUILD

GET http://127.0.0.1:8000/q3/search/location/Dadar Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Body Cookies Headers (9) Test Results Status: 200 OK Time: 16 ms Size: 586 B Save Re

Pretty Raw Preview Visualize JSON


```
[
    {
        "url": "http://127.0.0.1:8000/q3/list/restaurants/3/",
        "name": "Dominoes Pizza",
        "avgrating": 4,
        "description": "Domino's is the largest pizza restaurant chain in the world based on global retail sales.",
        "address": "Dadar, Mumbai",
        "cuisine": "Italian",
        "menu": [
            "http://127.0.0.1:8000/q3/list/dishes/2/"
        ]
    }
]
```


Search by cuisine:

Untitled Request

BUILD

GET http://127.0.0.1:8000/q3/search/cuisine/Italian Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Body Cookies Headers (9) Test Results Status: 200 OK Time: 15 ms Size: 890 B Save Res

Pretty Raw Preview Visualize JSON


```
[
    {
        "url": "http://127.0.0.1:8000/q3/list/restaurants/2/",
        "name": "Pizza Hut",
        "avgrating": 4,
        "description": "Pizza Hut is an American restaurant chain and international franchise founded in 1958.",
        "address": "Andheri, Mumbai",
        "cuisine": "Italian",
        "menu": [
            "http://127.0.0.1:8000/q3/list/dishes/2/"
        ]
    },
    {
        "url": "http://127.0.0.1:8000/q3/list/restaurants/3/",
        "name": "Dominoes Pizza",
        "avgrating": 4,
        "description": "Domino's is the largest pizza restaurant chain in the world based on global retail sales.",
        "address": "Dadar, Mumbai",
```


4.

Lab8/q4/

models.py

```
from django.db import models


class Amenity(models.Model):
    name = models.CharField(max_length=50)

    def __str__(self):
```

```
        return self.name


class AmenityProvider(models.Model):
    name = models.CharField(max_length=100)
    location = models.CharField(max_length=100)
    service = models.ForeignKey(Amenity, on_delete=models.CASCADE)
    avgRating = models.IntegerField()
    contactNumber = models.IntegerField()

    def __str__(self):
        return self.name


class Review(models.Model):
    review_by = models.CharField(max_length=100)
    text = models.CharField(max_length=1000)
    provider = models.ForeignKey(AmenityProvider,
on_delete=models.CASCADE)

    def __str__(self):
        return self.text
```

serializers.py

```
from rest_framework import serializers
from .models import Amenity, AmenityProvider, Review


class AmenitySerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = Amenity
        fields = '__all__'


class
AmenityProviderSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
        model = AmenityProvider
        fields = '__all__'


class ReviewSerializer(serializers.HyperlinkedModelSerializer):
    class Meta:
```

```
        model = Review
        fields = '__all__'
```

views.py

```
from rest_framework import viewsets
from .serializers import AmenitySerializer, AmenityProviderSerializer,
ReviewSerializer
from .models import Amenity, AmenityProvider, Review
from django.http.response import JsonResponse
from rest_framework.decorators import api_view

# viewset of rest framework takes care of crud operations


class amenity(viewsets.ModelViewSet):
    queryset = Amenity.objects.all()
    serializer_class = AmenitySerializer


class amenityProvider(viewsets.ModelViewSet):
    queryset = AmenityProvider.objects.all()
    serializer_class = AmenityProviderSerializer


class review(viewsets.ModelViewSet):
    queryset = Review.objects.all()
    serializer_class = ReviewSerializer

# functions for searching by location, name
@api_view(['GET'])
def getAmenityByLocation(request, location):
    amenities =
AmenityProvider.objects.filter(location__startswith=location)
    serializer = AmenitySerializer(
        amenities, context={'request': request}, many=True)
    return JsonResponse(serializer.data, safe=False)


@api_view(['GET'])
def getAmenityByName(request, name):
    amenities = Amenity.objects.filter(name__startswith=name)
    serializer = AmenitySerializer(
        amenities, context={'request': request}, many=True)
```

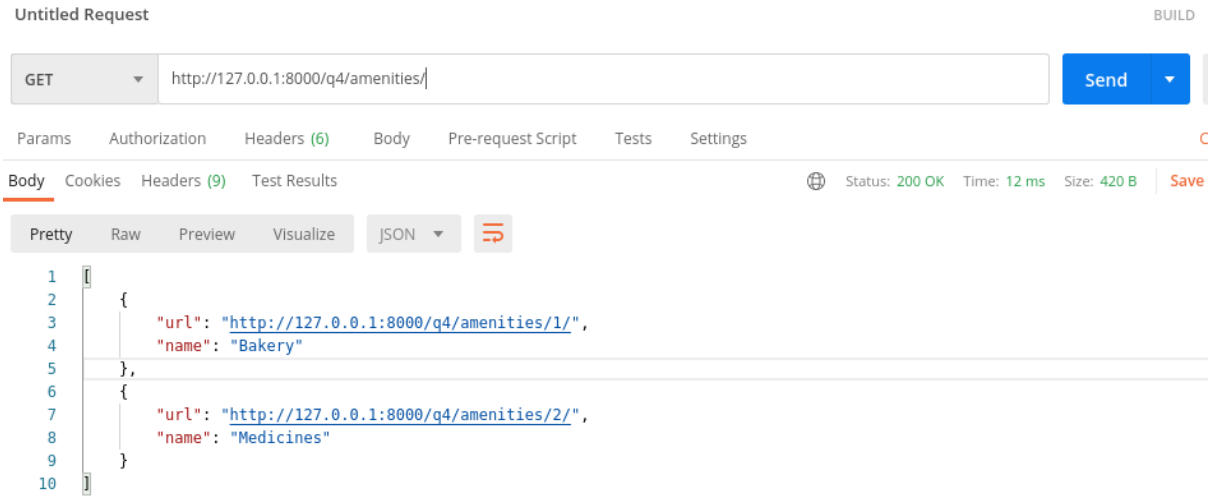
```
    return JsonResponse(serializer.data, safe=False)
```

urls.py

```
from django.urls import path, include
from . import views
from rest_framework import routers

router = routers.DefaultRouter()

router.register(r'amenities', views.amenity)
router.register(r'amenityProviders', views.amenityProvider)
router.register(r'reviewForProviders', views.review)

urlpatterns = [
    path('', include(router.urls)),
    path('search/location/<str:location>',
         views.getAmenityByLocation, name="getAmenityByLocation"),
    path('search/name/<str:name>', views.getAmenityByName,
         name="getAmenityByName"),
]
```

Screenshots:
CRUD operations available on amenities, amenity providers and reviews

Showing CRUD for amenities:
GET



POST

Untitled Request

BUILD

POST http://127.0.0.1:8000/q4/amenities/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "name":"Groceries"
3 }
```

Body Cookies Headers (10) Test Results Status: 201 Created Time: 224 ms Size: 407 B Save

Pretty Raw Preview Visualize JSON

```
1 {
2     "url": "http://127.0.0.1:8000/q4/amenities/3/",
3     "name": "Groceries"
4 }
```

PUT

PUT http://127.0.0.1:8000/q4/amenities/3/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2     "name":"Groceries and Vegetables"
3 }
```

Body Cookies Headers (9) Test Results Status: 200 OK Time: 316 ms Size: 382 B Save

Pretty Raw Preview Visualize JSON

```
1 {
2     "url": "http://127.0.0.1:8000/q4/amenities/3/",
3     "name": "Groceries and Vegetables"
4 }
```

DELETE
(Groceries and Vegetables)

DELETE http://127.0.0.1:8000/q4/amenities/3/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
{
    "name":"Groceries and Vegetables"
}
```

Body Cookies Headers (8) Test Results Status: 204 No Content Time: 313 ms Size: 276 B Save

Pretty Raw Preview Visualize Text

```
1
```

Showing GET on Amenity Providers:

GET http://127.0.0.1:8000/q4/amenityProviders/

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Body Cookies Headers (9) Test Results Status: 200 OK Tim

Pretty Raw Preview Visualize JSON


```
[
    {
        "url": "http://127.0.0.1:8000/q4/amenityProviders/1/",
        "name": "Hangout",
        "location": "Andheri",
        "avgRating": 3,
        "contactNumber": 8888888888,
        "service": "http://127.0.0.1:8000/q4/amenities/1/"
    },
    {
        "url": "http://127.0.0.1:8000/q4/amenityProviders/2/",
        "name": "Monginis",
        "location": "Dadar",
        "avgRating": 3,
        "contactNumber": 9999999999,
        "service": "http://127.0.0.1:8000/q4/amenities/1/"
    },
    {
        "url": "http://127.0.0.1:8000/q4/amenityProviders/3/",
        "name": "GetWell Medicines",
        "location": "Andheri",
        "avgRating": 4,
        "contactNumber": 3333333333,
        "service": "http://127.0.0.1:8000/q4/amenities/2/"
    }
]
```


Search Amenities by name:

GET http://127.0.0.1:8000/q4/search/name/Bakery Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Body Cookies Headers (9) Test Results Status: 200 OK Time: 13 ms Size: 343 B Sav

Pretty Raw Preview Visualize JSON


```
[
    {
        "url": "http://127.0.0.1:8000/q4/amenities/1/",
        "name": "Bakery"
    }
]
```


Search Amenity Providers by location:

GET http://127.0.0.1:8000/q4/search/location/Andheri

Params Authorization Headers (8) Body Pre-request Script Tests Setting

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON


```
[
    {
        "url": "http://127.0.0.1:8000/q4/amenities/1/",
        "name": "Hangout"
    },
    {
        "url": "http://127.0.0.1:8000/q4/amenities/3/",
        "name": "GetWell Medicines"
    }
]
```