IT Lab 8: REST API

Name: Arunima Singh Thakur
Roll Number: 31
Section: C
Batch: C1
Registration Number: 180905218

1. Create a ReST service for "ManipalBlog" with the following requirements:
• Users can register by providing email or phone number.
• Only registered users can create a new blog.
• Even anonymous users can comment on a blog.
Create HTTP methods for the following operations:
• User registration
• Update existing blog.
• Registered user adds comment.
• Anonymous user deletes comment
Test the service using POSTMAN.

manage.py:

```
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'week10v2.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
```

```
    main()
```

settings.py:

```
from pathlib import Path
import os
# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-6hv)wr9a8u3p4#rru)f)f3(^pf-$5g&i97#4m+ku$dry8$%y64'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['127.0.0.1']


# Application definition

INSTALLED_APPS = [
    'prob4.apps.Prob4Config',
    'prob3.apps.Prob3Config',
    'prob2.apps.Prob2Config',
    'prob1.apps.Prob1Config',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
```

```
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'week10v2.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'week10v2.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'week10',
        'USER': 'itlabuser',
        'PASSWORD': 'incorrect',
        'HOST': 'localhost',

    }
}


# Password validation
# https://docs.djangoproject.com/en/3.2/ref/settings/#auth-password-validators
```

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/3.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC_URL = '/static/'

# Default primary key field type
# https://docs.djangoproject.com/en/3.2/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'
```

models.py:

```
from re import T
```

```python
from django.db import models
# Create your models here.
class User(models.Model):
    username = models.CharField(unique=True,null=False,blank=False,max_length=200)
    email = models.EmailField(null = True,blank=True)
    phno = models.PositiveBigIntegerField(null=True,blank=True)
    password = models.CharField(null=False,blank=False,max_length=200)
    def __str__(self):
        return self.username

class Blog(models.Model):
    title = models.CharField(max_length=200)
    desc = models.TextField()
    date = models.DateField()
    user = models.ForeignKey(User,on_delete=models.CASCADE,default = None)
    def __str__(self):
        return self.title

class Comment(models.Model):
    user = models.ForeignKey(User,on_delete=models.CASCADE,null=True)
    Blog = models.ForeignKey(Blog,on_delete=models.CASCADE)
    comment = models.TextField()
    date = models.DateField()
```

serializers.py:

```python
from django.db.models import fields
from rest_framework import serializers
from .models import Comment,Blog,User


class UserSerializer(serializers.ModelSerializer):
    class Meta:
        fields = (
            'id',
            'username',
            'email',
            'phno',
            'password',
        )
        model = User
```

```
class CommentSerializer(serializers.ModelSerializer):
   class Meta:
       fields = (
           'id',
           'user',
           'Blog',
           'comment',
           'date'
       )
       model = Comment

class BlogSerializer(serializers.ModelSerializer):
   class Meta:
       fields = (
           'id',
           'title',
           'desc',
           'date',
           'user',
       )
       model = Blog
```

views.py:

```
from django.shortcuts import render
from django.http import request
from .models import *
from .serializers import *
from rest_framework import generics
import getpass

class ListBlogs(generics.ListCreateAPIView):
   queryset = Blog.objects.all()
   serializer_class = BlogSerializer

class DetailBlog(generics.RetrieveUpdateDestroyAPIView):
   queryset = Blog.objects.all()
   serializer_class = BlogSerializer

class ListComment(generics.ListCreateAPIView):
   queryset = Comment.objects.all()
   serializer_class = CommentSerializer
```

```
class DetailComment(generics.RetrieveUpdateDestroyAPIView):
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer

class ListUser(generics.ListCreateAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer

class DetailUser(generics.RetrieveUpdateDestroyAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer
```

urls.py:

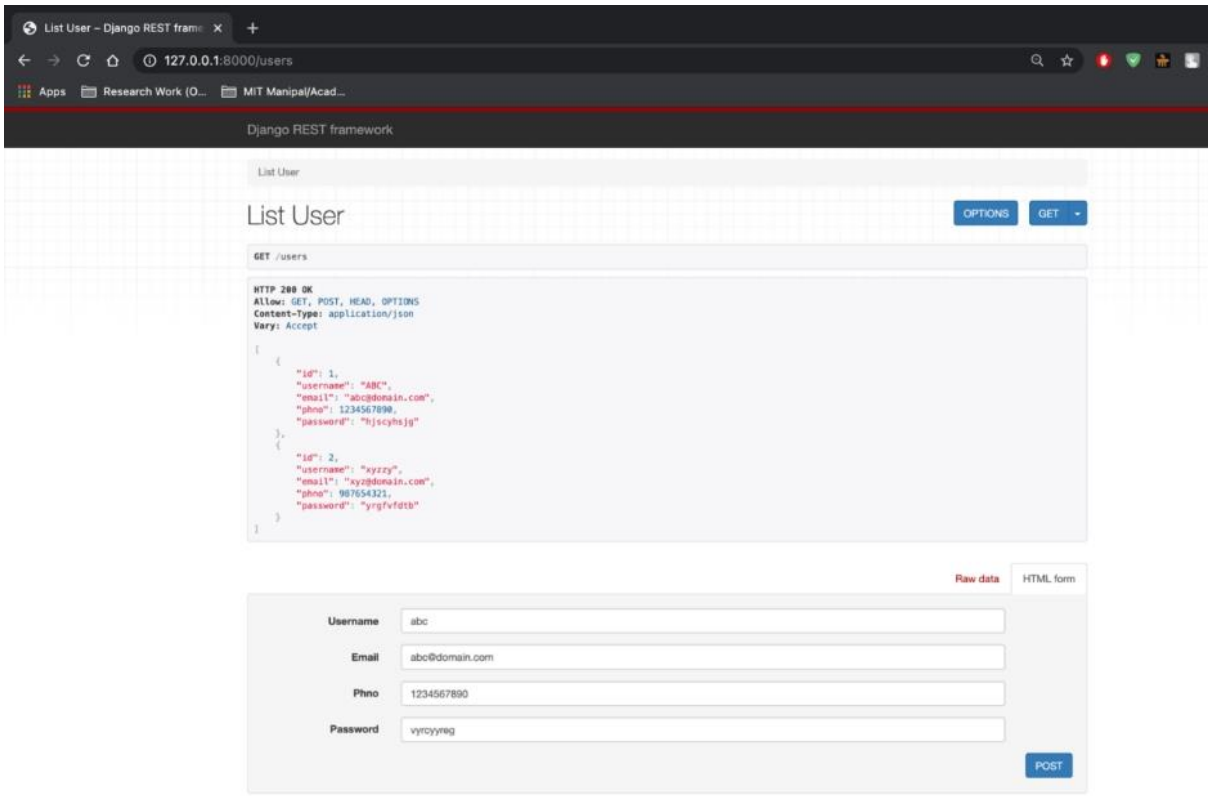
```
from django.urls import path
from .views import *
urlpatterns = [
    path("blogs",ListBlogs.as_view(),name = "ListBlog"),
    path("blogs/<int:pk>",DetailBlog.as_view(),name = "Blog"),
    path("comments",ListComment.as_view(),name="comments"),
    path("users",ListUser.as_view(),name = "users"),
    path("users/<int:pk>",DetailUser.as_view(),name = "User"),
    path("comments/<int:pk>",DetailComment.as_view(),name="comment"),
]
```

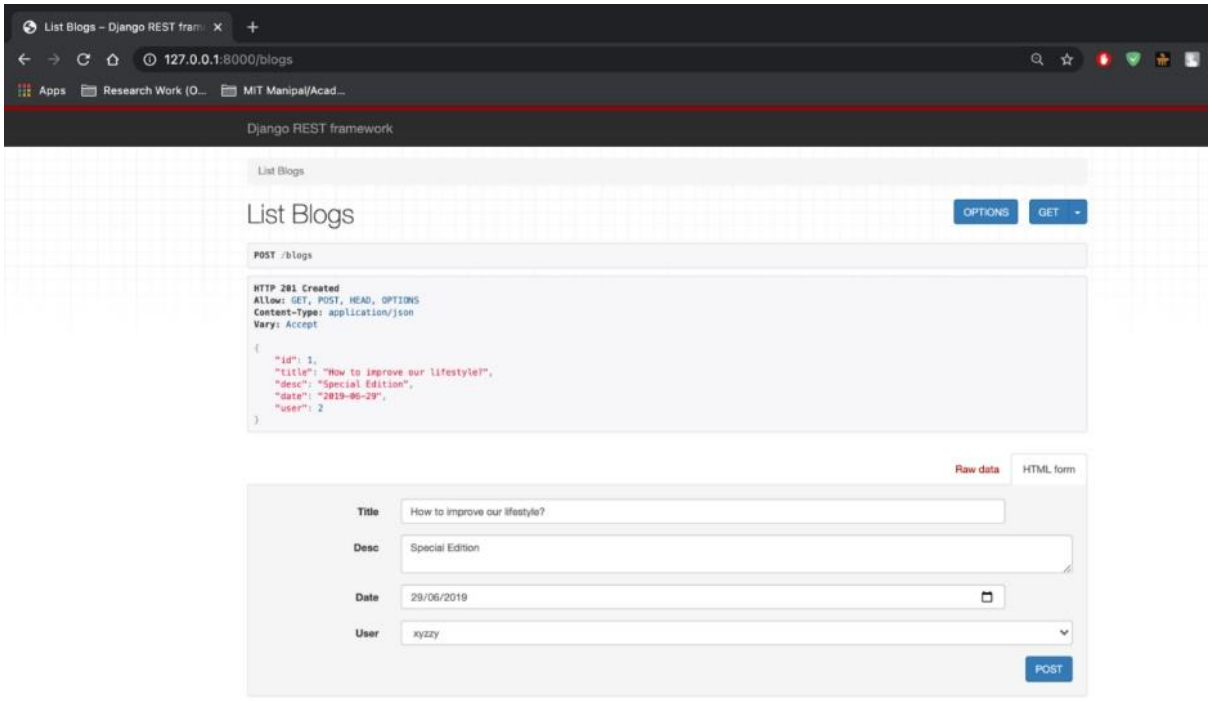
urls.py:

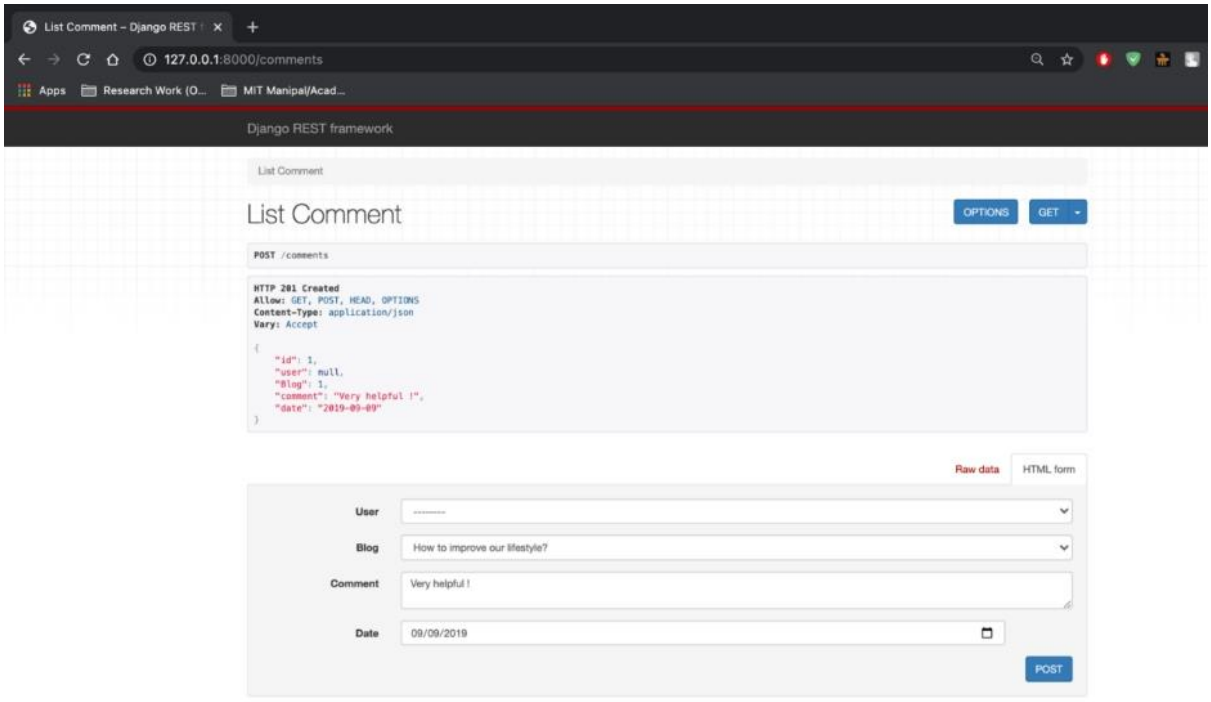
```
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('',include("prob1.urls")),
    #path('',include("prob2.urls")),
    #path('',include("prob3.urls")),
    #path('',include("prob4.urls")),
```

```
]
```

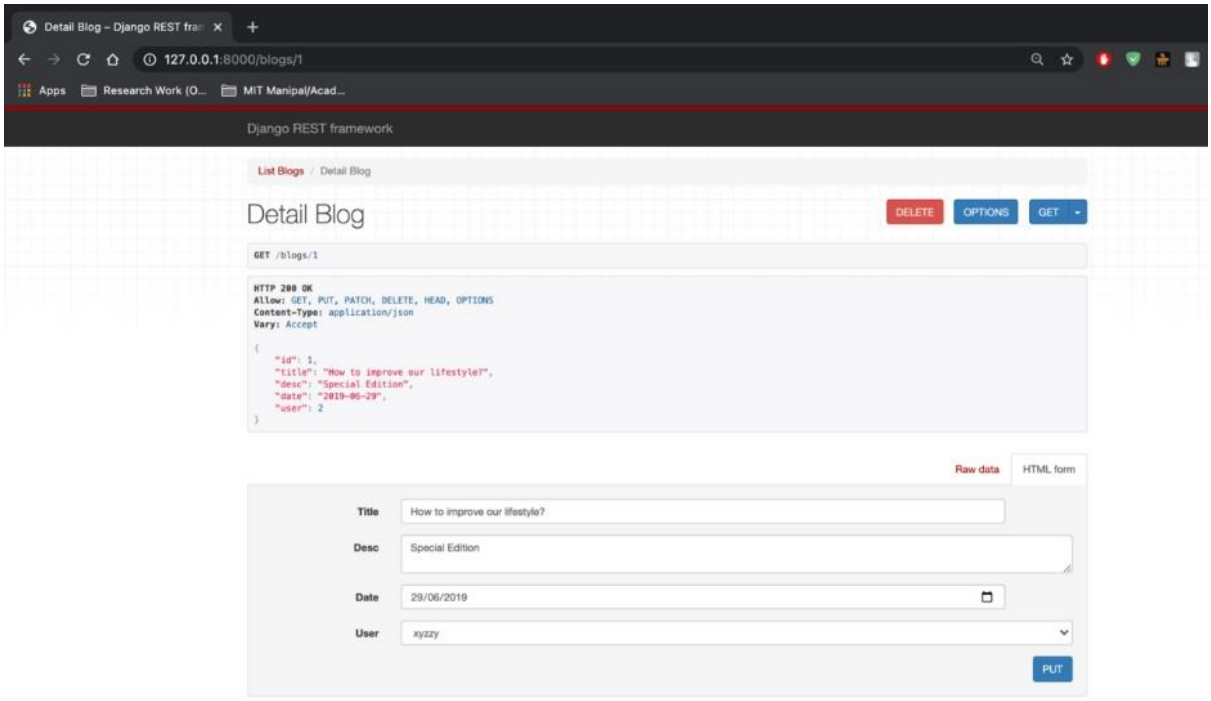
Output:

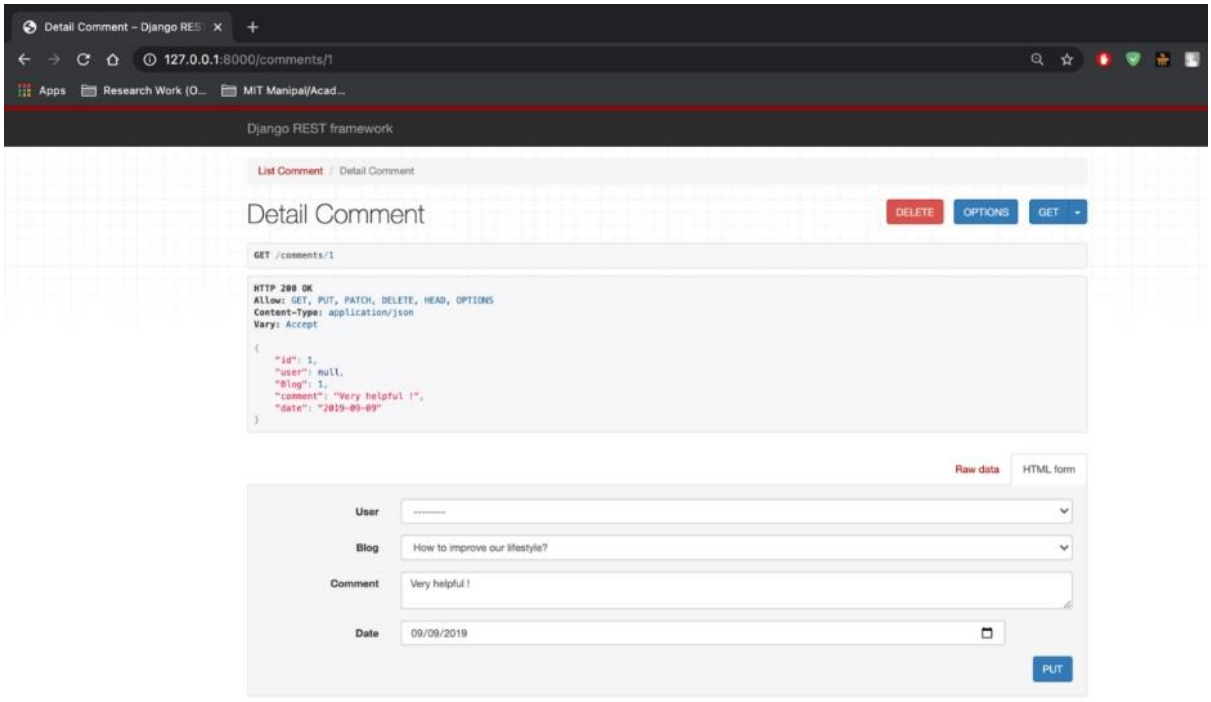

List User – Django REST frame
127.0.0.1:8000/users
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List User
List User
OPTIONS
GET
GET /users
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"id": 1,
"username": "ABC",
"email": "abc@domain.com",
"phno": 1234567890,
"password": "hjscyhsjg"
},
{
"id": 2,
"username": "xyzzy",
"email": "xyz@domain.com",
"phno": 987654321,
"password": "yrgfvfdtb"
}
]
Raw data
HTML form
Username
abc
Email
abc@domain.com
Phno
1234567890
Password
vyrcyyreg
POST


List Blogs – Django REST fram
127.0.0.1:8000/blogs
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Blogs
List Blogs
OPTIONS
GET
POST /blogs
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"id": 1,
"title": "How to improve our lifestyle?",
"desc": "Special Edition",
"date": "2019-06-29",
"user": 2
}
Raw data
HTML form
Title
How to improve our lifestyle?
Desc
Special Edition
Date
29/06/2019
User
xyzzy
POST


List Comment – Django REST
127.0.0.1:8000/comments
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Comment
List Comment
OPTIONS
GET
POST /comments
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"id": 1,
"user": null,
"Blog": 1,
"comment": "Very helpful !",
"date": "2019-09-09"
}
Raw data
HTML form
User

Blog
How to improve our lifestyle?
Comment
Very helpful !
Date
09/09/2019
POST


Detail Blog – Django REST fra
127.0.0.1:8000/blogs/1
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Blogs / Detail Blog
Detail Blog
DELETE
OPTIONS
GET
GET /blogs/1
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"id": 1,
"title": "How to improve our lifestyle?",
"desc": "Special Edition",
"date": "2019-06-29",
"user": 2
}
Raw data
HTML form
Title
How to improve our lifestyle?
Desc
Special Edition
Date
29/06/2019
User
xyzzy
PUT


Detail Comment – Django RES
127.0.0.1:8000/comments/1
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Comment / Detail Comment
Detail Comment
DELETE
OPTIONS
GET
GET /comments/1
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"id": 1,
"user": null,
"Blog": 1,
"comment": "Very helpful !",
"date": "2019-09-09"
}
Raw data
HTML form
User

Blog
How to improve our lifestyle?
Comment
Very helpful !
Date
09/09/2019
PUT

2. Create a ReST service for Ola Cabs with the requirement given below:
The service should provide the following real time information about Ola rides available
at a given user location (latitude and longitude).
• Estimated time of arrival (ETA)
• Fare details
Implement the CRUD operations for the resources identified and create a client to consume the service.

models.py:

```
from django.db import models

# Create your models here.
class UserData(models.Model):
    name = models.CharField(max_length=100)
    contact = models.PositiveBigIntegerField()
    def __str__(self):
        return self.name

class UserLocation(models.Model):
    user = models.ForeignKey(UserData,on_delete=models.CASCADE)
    latitude = models.DecimalField(max_digits=7,decimal_places=4)
    longitude = models.DecimalField(max_digits=7,decimal_places=4)
    def __str__(self):
        return '{}_{}_{}'.format(self.latitude,self.longitude,self.user.name)
```

```
class VehicleInfo(models.Model):
    driverName = models.CharField(max_length = 100)
    vehicleName = models.CharField(max_length = 100)
    vehicleRegNo = models.CharField(max_length=10)
    contact = models.PositiveBigIntegerField()

    def __str__(self):
        return self.driverName+"_"+self.vehicleName+"_"+self.vehicleRegNo


class TravelStatus(models.Model):
    userLocation = models.ForeignKey(UserLocation,on_delete=models.CASCADE)
    vehicle = models.ForeignKey(VehicleInfo,on_delete=models.CASCADE)
    eta = models.TimeField()
    fare = models.PositiveIntegerField()
```

serializers.py:

```
from django.db.models import fields
from rest_framework import serializers
from .models import *

class UserDataserializer(serializers.ModelSerializer):
    class Meta:
        fields = '__all__'
        model = UserData

class UserLocationserializer(serializers.ModelSerializer):
    class Meta:
        fields = '__all__'
        model = UserLocation

class VehicleInfoserializer(serializers.ModelSerializer):
    class Meta:
        fields = '__all__'
        model = VehicleInfo

class TravelStatusserializer(serializers.ModelSerializer):
    class Meta:
        fields = '__all__'
        model = TravelStatus
```

views.py:

```
from django.shortcuts import render
from .serializers import *
from .models import *
from rest_framework import generics
# Create your views here.
class ListUserData(generics.ListCreateAPIView):
    queryset = UserData.objects.all()
    serializer_class = UserDataserializer

class DetailUserData(generics.RetrieveUpdateDestroyAPIView):
    queryset = UserData.objects.all()
    serializer_class = UserDataserializer

class ListUserLocation(generics.ListCreateAPIView):
    queryset = UserLocation.objects.all()
    serializer_class = UserLocationserializer

class DetailUserLocation(generics.RetrieveUpdateDestroyAPIView):
    queryset = UserLocationserializer
    serializer_class = UserLocationserializer

class ListVehicleInfo(generics.ListCreateAPIView):
    queryset = VehicleInfo.objects.all()
    serializer_class = VehicleInfoserializer

class DetailVehicleInfo(generics.RetrieveUpdateDestroyAPIView):
    queryset = VehicleInfo.objects.all()
    serializer_class = VehicleInfoserializer

class ListTravelStatus(generics.ListCreateAPIView):
    queryset = TravelStatus.objects.all()
    serializer_class = TravelStatusserializer

class DetailTravelStatus(generics.RetrieveUpdateDestroyAPIView):
    queryset = TravelStatus.objects.all()
    serializer_class = TravelStatusserializer
```

urls.py:

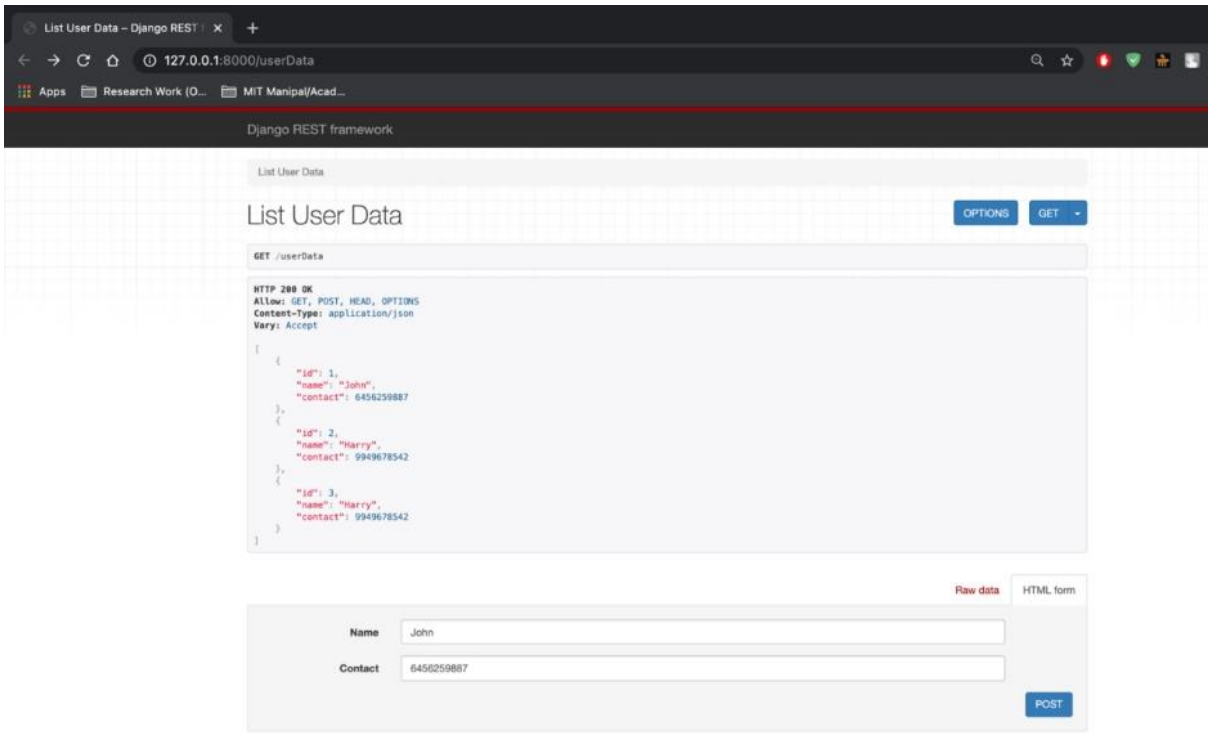
```
from django.urls import path
```

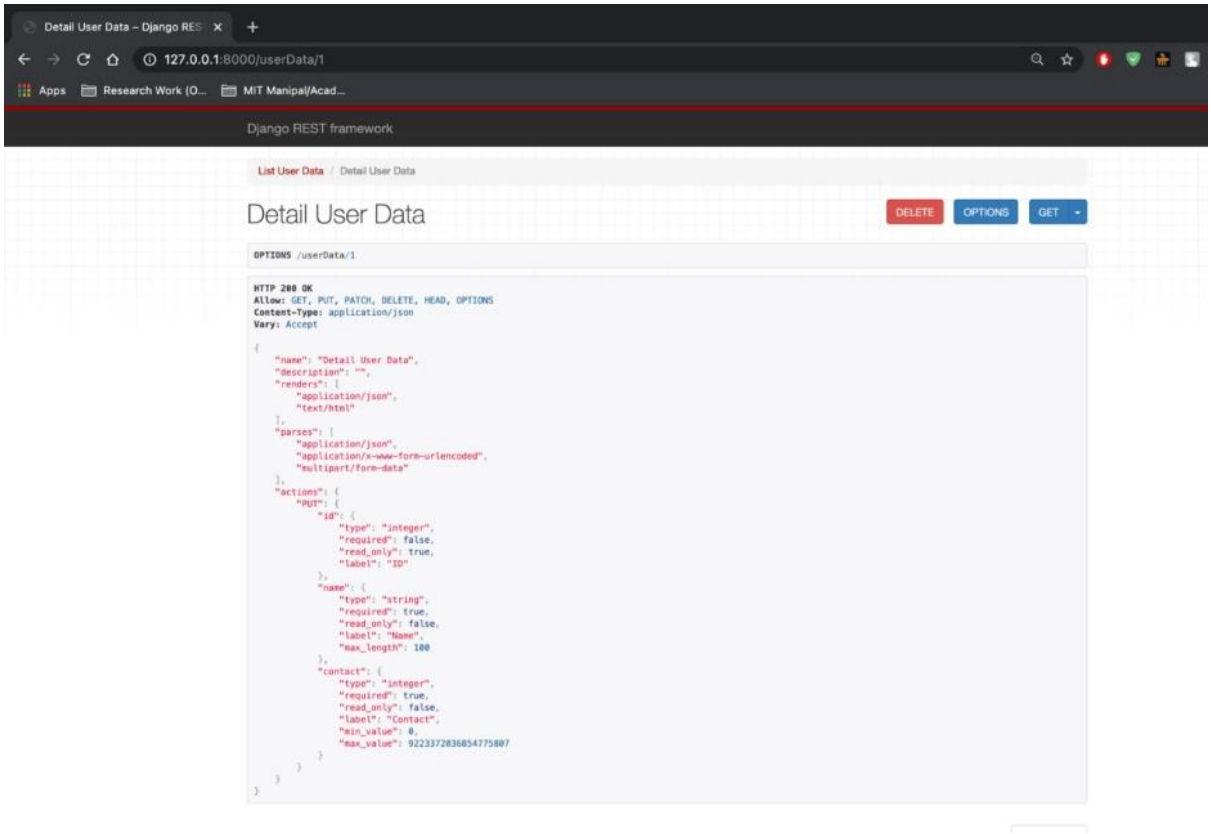
```
from django.urls.resolvers import URLPattern
from .views import *

urlpatterns = [
   path("userData",ListUserData.as_view(),name = "userData"),
   path("userLocation",ListUserLocation.as_view(),name = "usersLocation"),
   path("vehicleInfo",ListVehicleInfo.as_view(),name = "vehiclesInfo"),
   path("travelStatus",ListTravelStatus.as_view(),name = "travelStatuses"),
   path("userData/<int:pk>",DetailUserData.as_view(),name = "userDatum"),
   path("userLocation/<int:pk>",DetailUserLocation.as_view(),name =
"userLocation"),
   path("vehicleInfo/<int:pk>",DetailVehicleInfo.as_view(),name = "vehicleInfo"),
   path("travelStatus/<int:pk>",DetailTravelStatus.as_view(),name =
"travelStatus"),
]
```

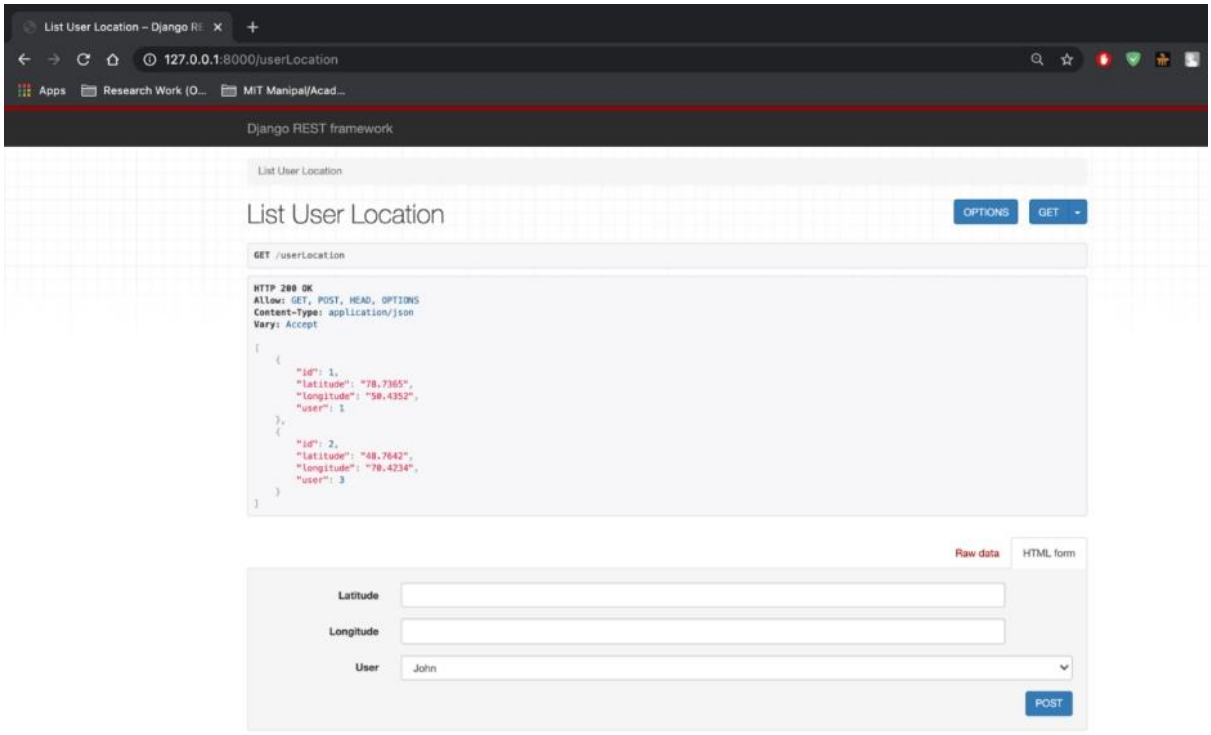
urls.py:

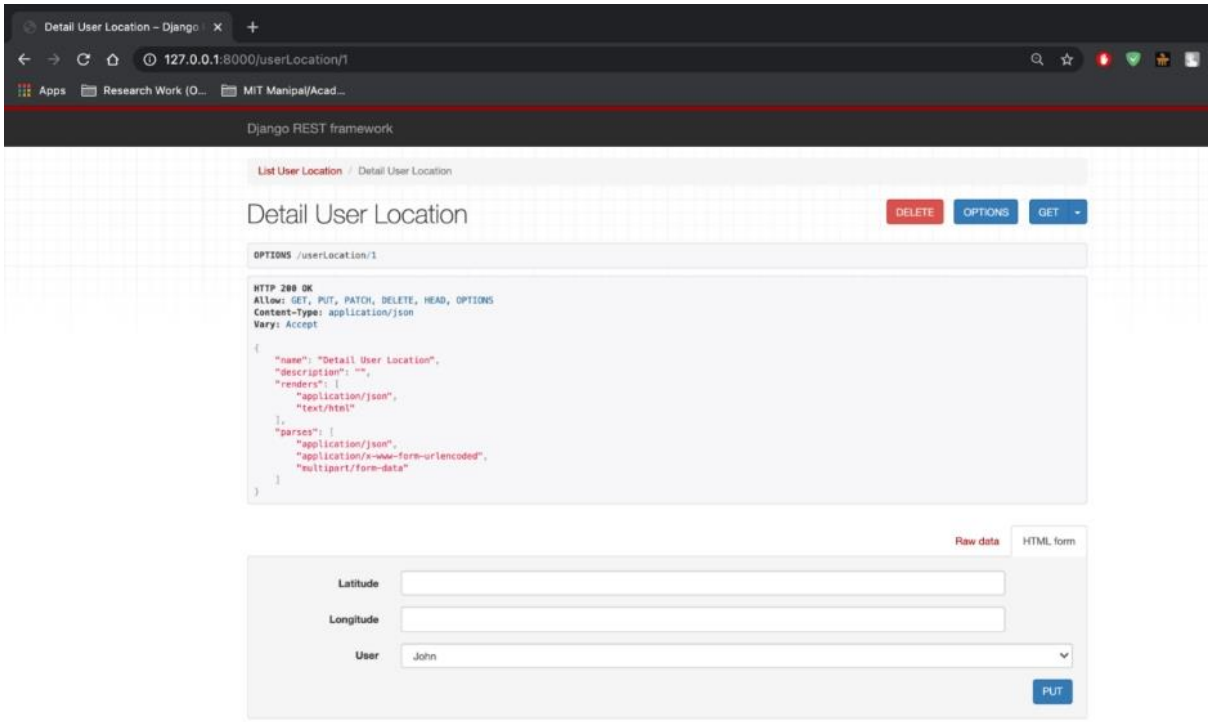
```
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
   path('admin/', admin.site.urls),
   #path('',include("prob1.urls")),
   path('',include("prob2.urls")),
   #path('',include("prob3.urls")),
   #path('',include("prob4.urls")),
]
```

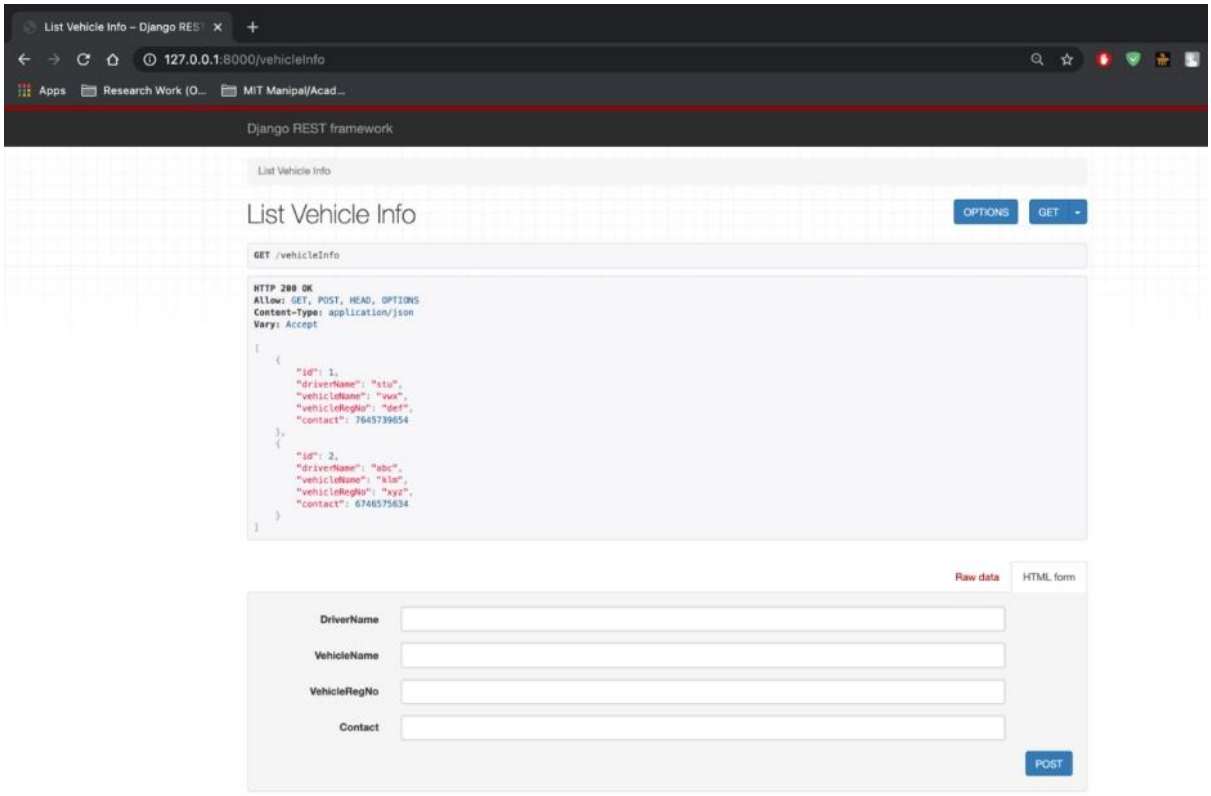
Output:

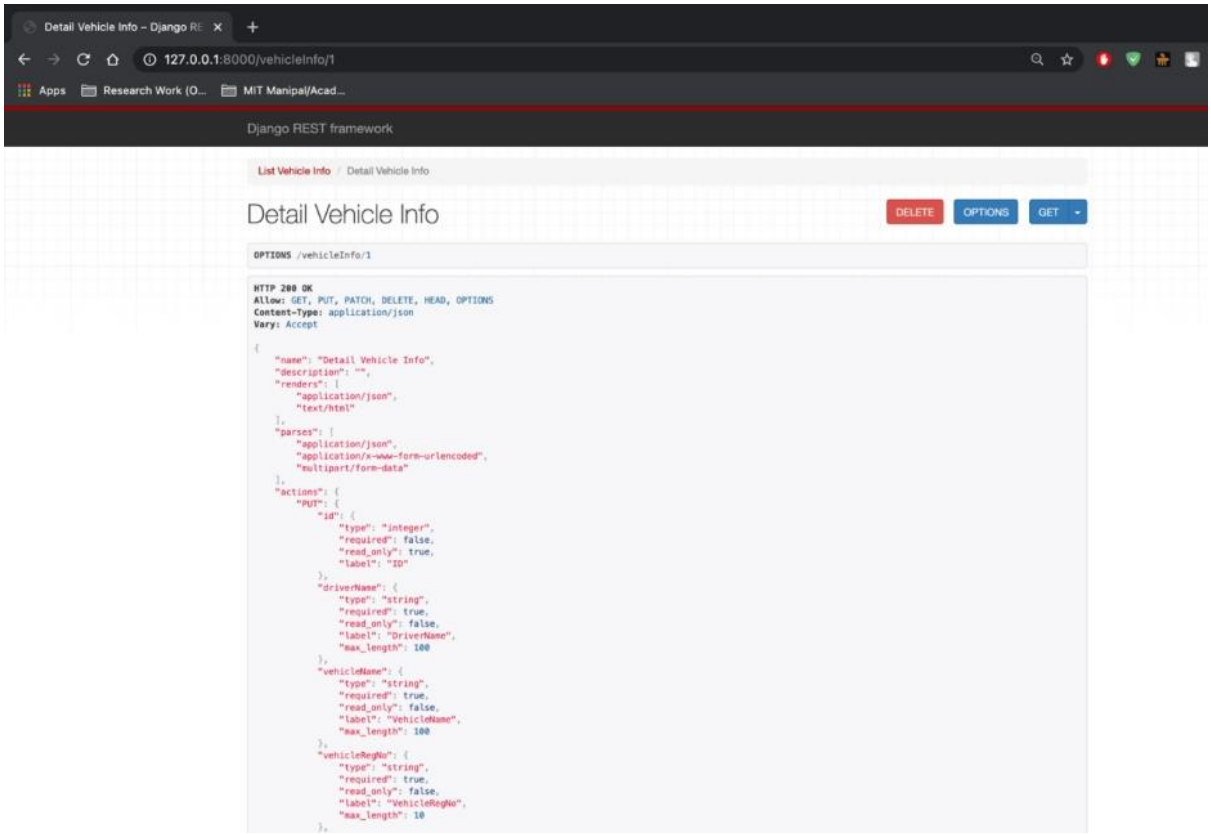

List User Data – Django REST
127.0.0.1:8000/userData
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List User Data
List User Data
OPTIONS
GET
GET /userData
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"id": 1,
"name": "John",
"contact": 6456259887
},
{
"id": 2,
"name": "Harry",
"contact": 9949678542
},
{
"id": 3,
"name": "Harry",
"contact": 9949678542
}
]
Raw data
HTML form
Name
John
Contact
6456259887
POST

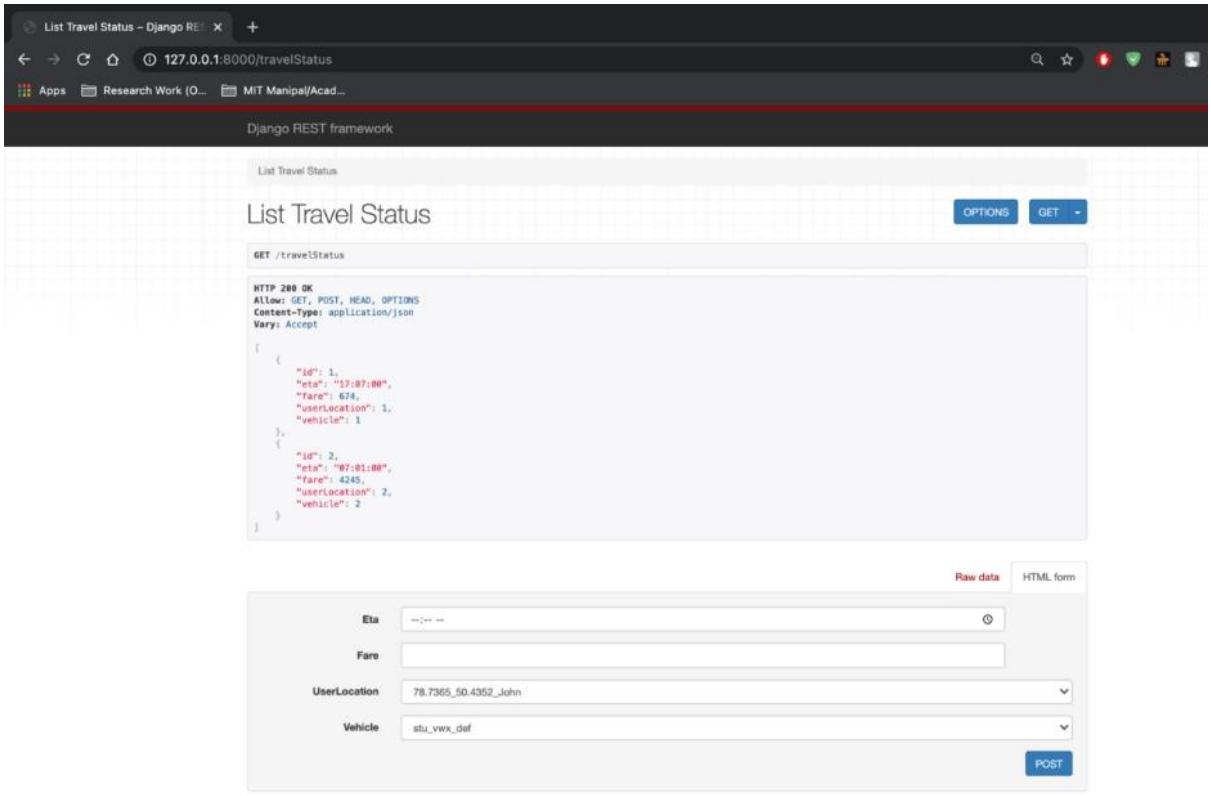

Detail User Data – Django RES
127.0.0.1:8000/userData/1
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List User Data / Detail User Data
Detail User Data
DELETE
OPTIONS
GET
OPTIONS /userData/1
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"name": "Detail User Data",
"description": "",
"renders": [
"application/json",
"text/html"
],
"parses": [
"application/json",
"application/x-www-form-urlencoded",
"multipart/form-data"
],
"actions": {
"PUT": {
"id": {
"type": "integer",
"required": false,
"read_only": true,
"label": "ID"
},
"name": {
"type": "string",
"required": true,
"read_only": false,
"label": "Name",
"max_length": 100
},
"contact": {
"type": "integer",
"required": true,
"read_only": false,
"label": "Contact",
"min_value": 0,
"max_value": 9223372036854775807
}
}
}
}

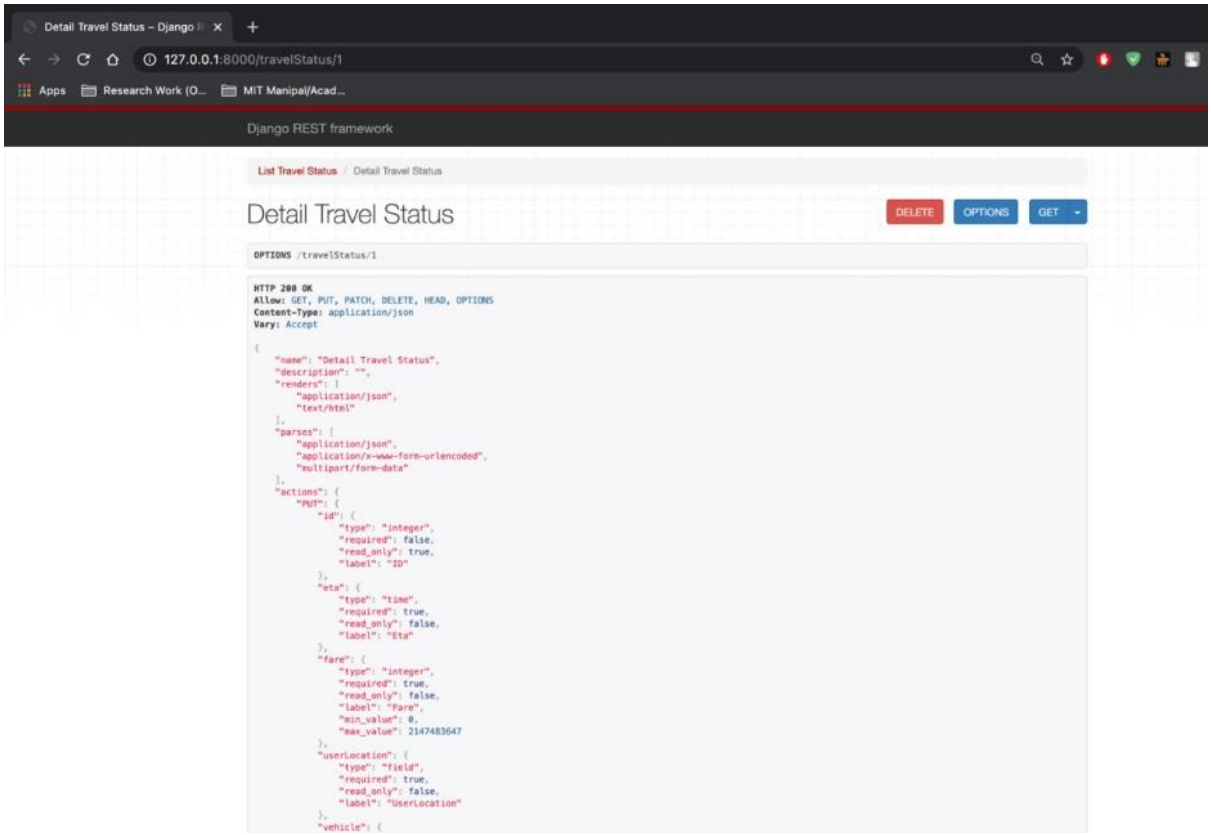

List User Location – Django RE
127.0.0.1:8000/userLocation
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List User Location
List User Location
OPTIONS
GET
GET /userLocation
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"id": 1,
"latitude": "78.7365",
"longitude": "50.4352",
"user": 1
},
{
"id": 2,
"latitude": "48.7642",
"longitude": "70.4234",
"user": 3
}
]
Raw data
HTML form
Latitude
Longitude
User
John
POST


Detail User Location – Django
127.0.0.1:8000/userLocation/1
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List User Location / Detail User Location
Detail User Location
DELETE
OPTIONS
GET
OPTIONS /userLocation/1
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"name": "Detail User Location",
"description": "",
"renders": [
"application/json",
"text/html"
],
"parses": [
"application/json",
"application/x-www-form-urlencoded",
"multipart/form-data"
]
}
Raw data
HTML form
Latitude
Longitude
User
John
PUT


List Vehicle Info – Django RES
127.0.0.1:8000/vehicleInfo
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Vehicle Info
List Vehicle Info
OPTIONS
GET
GET /vehicleInfo
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"id": 1,
"driverName": "stu",
"vehicleName": "vwx",
"vehicleRegNo": "def",
"contact": 7645739654
},
{
"id": 2,
"driverName": "abc",
"vehicleName": "klm",
"vehicleRegNo": "xyz",
"contact": 6746575634
}
]
Raw data
HTML form
DriverName
VehicleName
VehicleRegNo
Contact
POST


Detail Vehicle Info – Django RE
127.0.0.1:8000/vehicleInfo/1
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Vehicle Info / Detail Vehicle Info
Detail Vehicle Info
DELETE
OPTIONS
GET
OPTIONS /vehicleInfo/1
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"name": "Detail Vehicle Info",
"description": "",
"renders": [
"application/json",
"text/html"
],
"parses": [
"application/json",
"application/x-www-form-urlencoded",
"multipart/form-data"
],
"actions": {
"PUT": {
"id": {
"type": "integer",
"required": false,
"read_only": true,
"label": "ID"
},
"driverName": {
"type": "string",
"required": true,
"read_only": false,
"label": "DriverName",
"max_length": 100
},
"vehicleName": {
"type": "string",
"required": true,
"read_only": false,
"label": "VehicleName",
"max_length": 100
},
"vehicleRegNo": {
"type": "string",
"required": true,
"read_only": false,
"label": "VehicleRegNo",
"max_length": 10
},


List Travel Status – Django RE
127.0.0.1:8000/travelStatus
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Travel Status
List Travel Status
OPTIONS
GET
GET /travelStatus
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"id": 1,
"eta": "17:07:00",
"fare": 674,
"userLocation": 1,
"vehicle": 1
},
{
"id": 2,
"eta": "07:01:00",
"fare": 4245,
"userLocation": 2,
"vehicle": 2
}
]
Raw data
HTML form
Eta
--:-- --
Fare
UserLocation
78.7365_50.4352_John
Vehicle
stu_vwx_def
POST


Detail Travel Status – Django R
127.0.0.1:8000/travelStatus/1
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Travel Status / Detail Travel Status
Detail Travel Status
DELETE
OPTIONS
GET
OPTIONS /travelStatus/1
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"name": "Detail Travel Status",
"description": "",
"renders": [
"application/json",
"text/html"
],
"parses": [
"application/json",
"application/x-www-form-urlencoded",
"multipart/form-data"
],
"actions": {
"PUT": {
"id": {
"type": "integer",
"required": false,
"read_only": true,
"label": "ID"
},
"eta": {
"type": "time",
"required": true,
"read_only": false,
"label": "Eta"
},
"fare": {
"type": "integer",
"required": true,
"read_only": false,
"label": "Fare",
"min_value": 0,
"max_value": 2147483647
},
"userLocation": {
"type": "field",
"required": true,
"read_only": false,
"label": "UserLocation"
},
"vehicle": {

3. Design and implement a ReST service for Romato, which gives you access to the freshest and most exhaustive information for over 1 million restaurants across 1,000 cities globally. With the Romato APIs, one can search for restaurants by name, cuisine, orlocation. Identify any three resources and implement CRUD operations.

models.py:

```
from django.db import models
# Create your models here.

class Customer(models.Model):
    name = models.CharField(max_length=100)
    contact = models.PositiveBigIntegerField()

class Staff(models.Model):
    name = models.CharField(max_length=100)
    designation = models.CharField(max_length=200)
    contact = models.PositiveBigIntegerField()

class Restaurant(models.Model):
    name = models.CharField(max_length=200)
    cuisine = models.CharField(max_length=100)
    location = models.CharField(max_length=100)
    contact = models.PositiveBigIntegerField()
```

serializers.py:

```
from django.db.models import fields
from rest_framework import serializers
from .models import *

class CustomerSerializer(serializers.ModelSerializer):
    class Meta:
        fields = '__all__'
        model = Customer

class StaffSerializer(serializers.ModelSerializer):
    class Meta:
        fields = '__all__'
        model = Staff

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

views.py:

```
from django.shortcuts import render
from rest_framework import generics,filters
from .serializers import *
from .models import *
# Create your views here
class ListCustomer(generics.ListCreateAPIView):
    queryset = Customer.objects.all()
    serializer_class = CustomerSerializer

class DetailCustomer(generics.RetrieveUpdateDestroyAPIView):
    queryset = Customer.objects.all()
    serializer_class = CustomerSerializer

class ListStaff(generics.ListCreateAPIView):
    queryset = Staff.objects.all()
    serializer_class = StaffSerializer

class DetailStaff(generics.RetrieveUpdateDestroyAPIView):
    queryset = Staff.objects.all()
    serializer_class = StaffSerializer

class ListRestaurant(generics.ListCreateAPIView):
    queryset = Restaurant.objects.all()
    serializer_class = RestaurantSerializer
```

```
    filter_backends = [filters.SearchFilter]
    search_fields = ['name','cuisine','location']

class DetailRestaurant(generics.RetrieveUpdateDestroyAPIView):
    queryset = Restaurant.objects.all()
    serializer_class = RestaurantSerializer
```

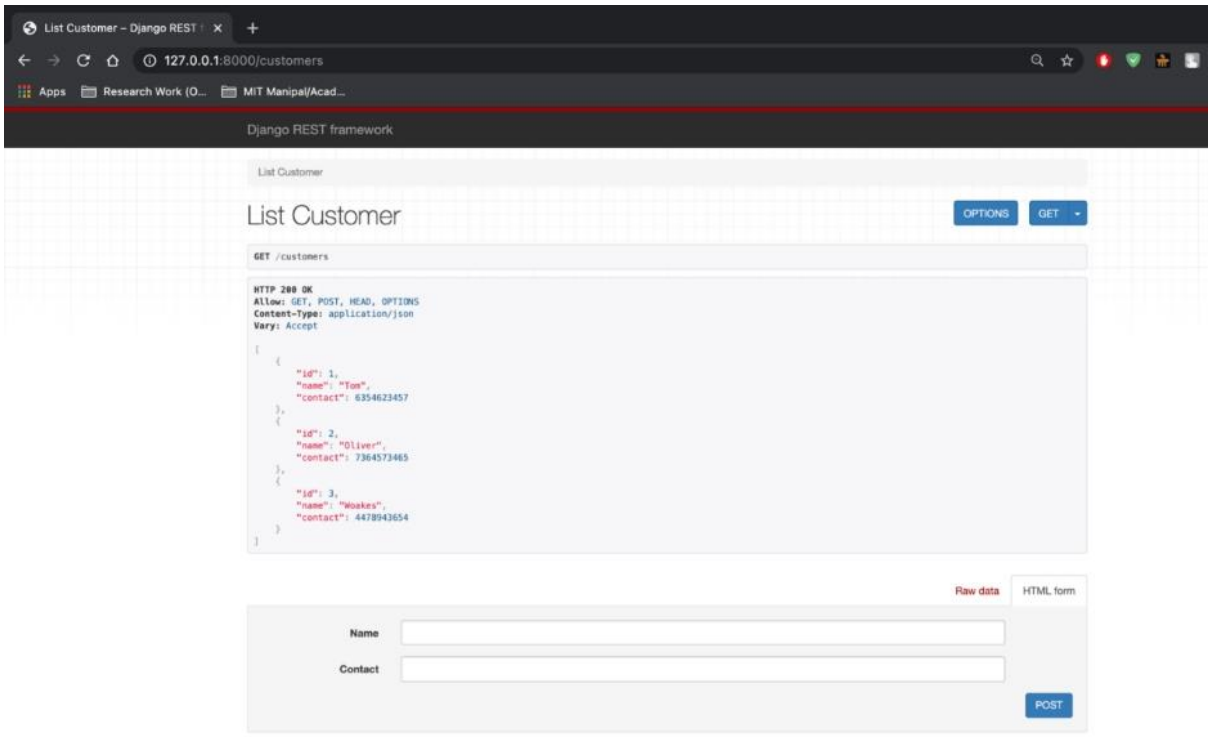
urls.py:

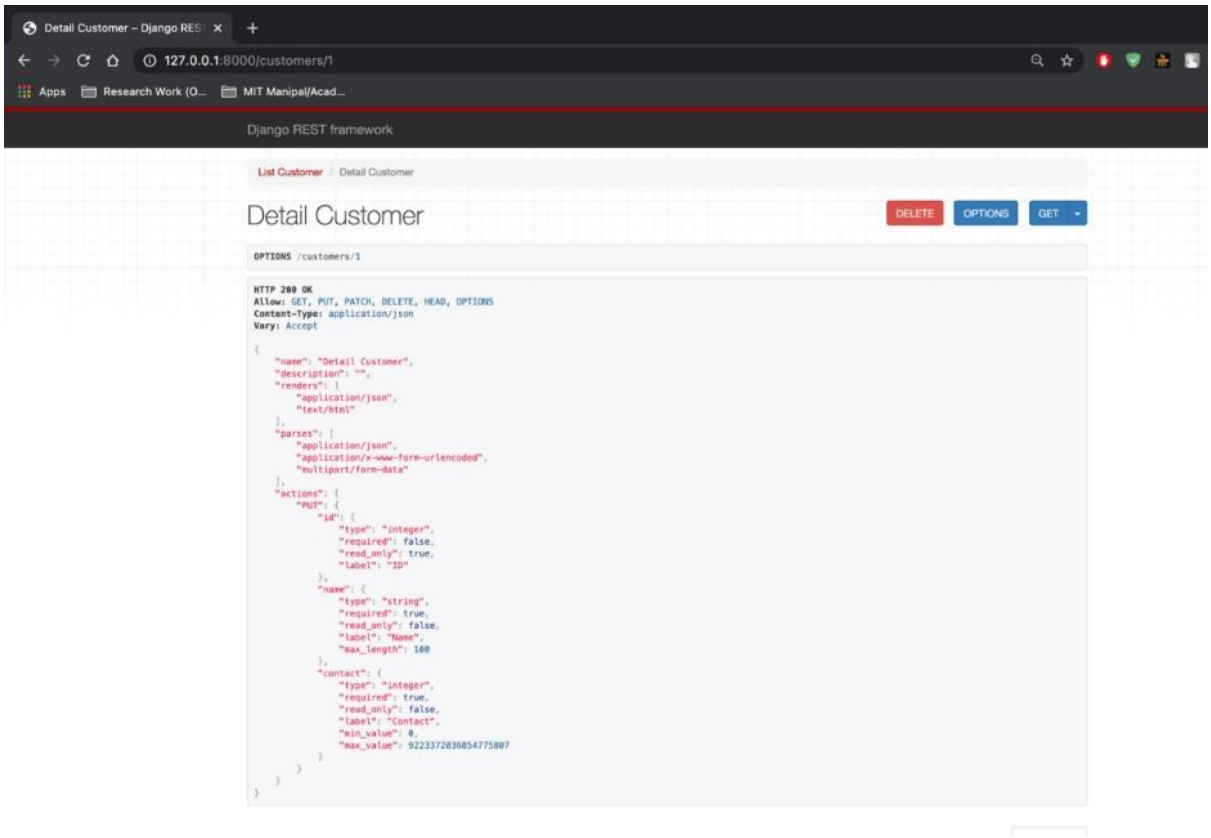
```
from django.urls import path
from .views import *

urlpatterns = [
    path("customers",ListCustomer.as_view(),name = "customers"),
    path("staff",ListStaff.as_view(),name = "staffs"),
    path("restaurants",ListRestaurant.as_view(),name = "restaurants"),
    path("customers/<int:pk>",DetailCustomer.as_view(),name = "customer"),
    path("staff/<int:pk>",DetailStaff.as_view(),name = "staff"),
    path("restaurants/<int:pk>",DetailRestaurant.as_view(),name = "restaurant"),
]
```

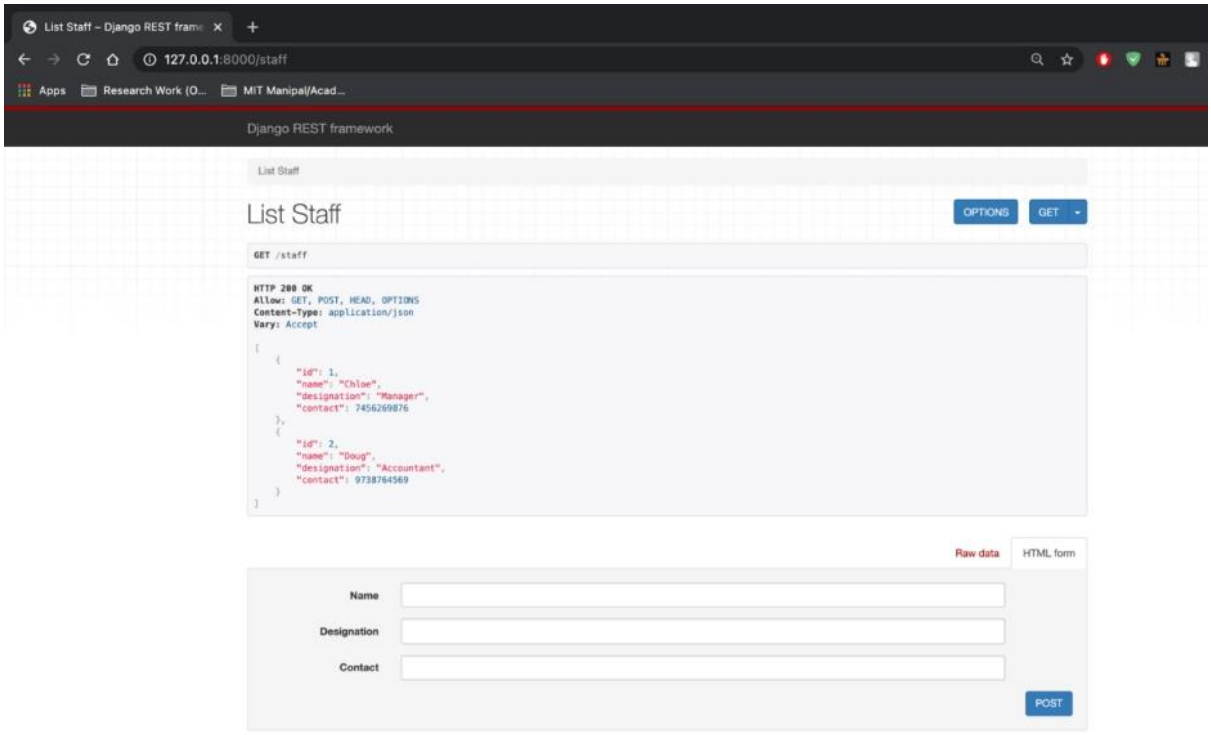
urls.py:

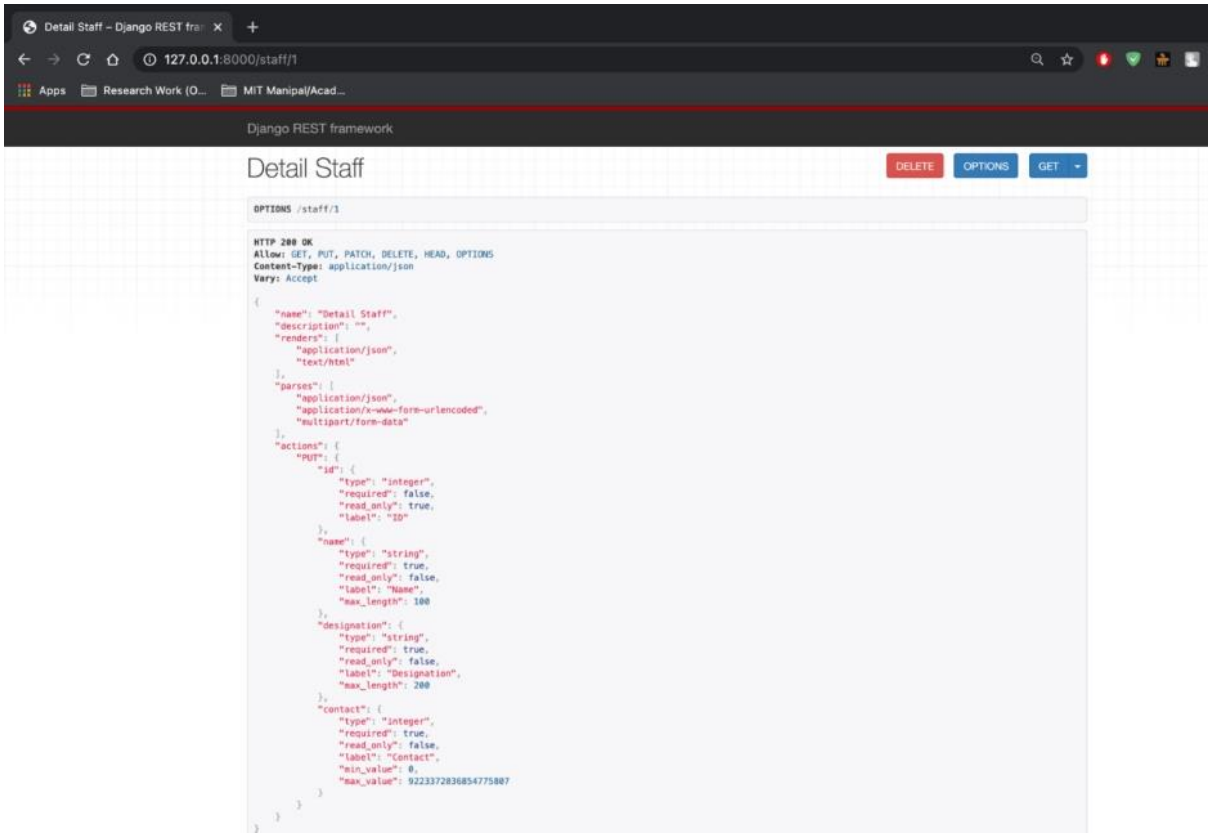
```
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    #path('',include("prob1.urls")),
    #path('',include("prob2.urls")),
    path('',include("prob3.urls")),
    #path('',include("prob4.urls")),
]
```

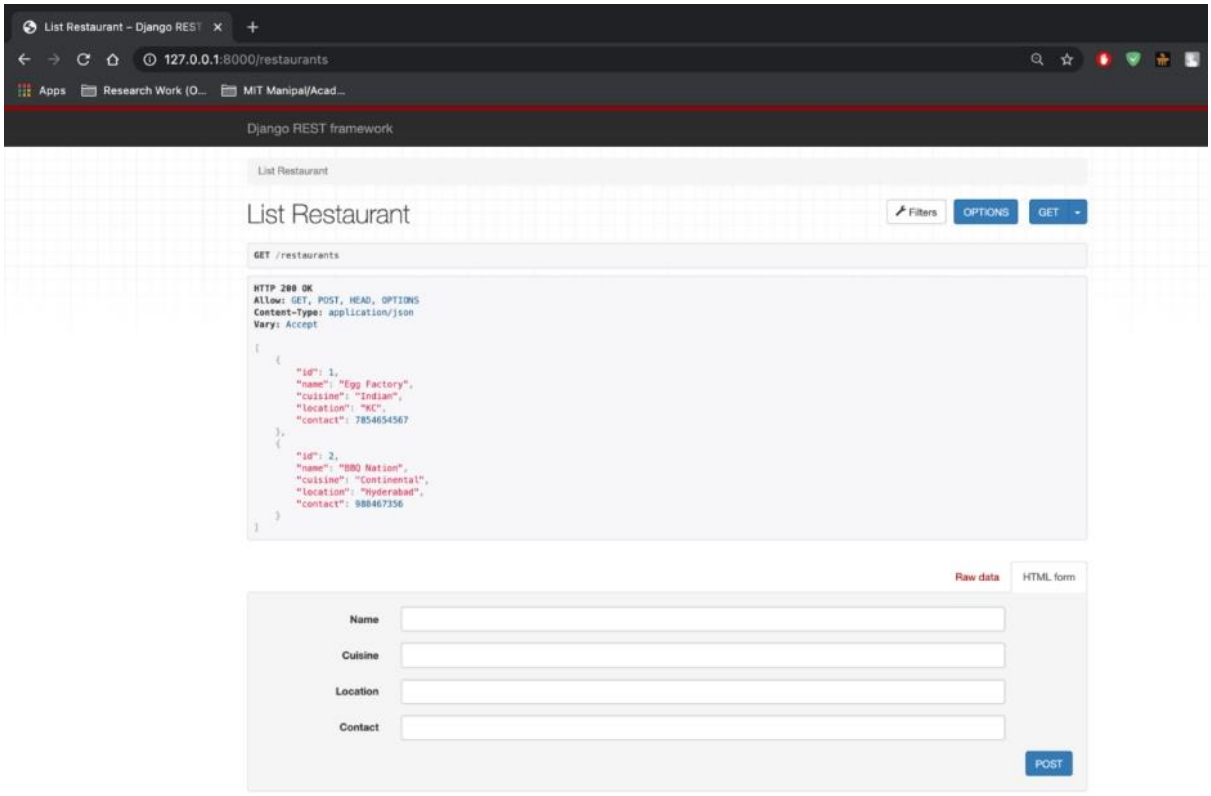
Output:

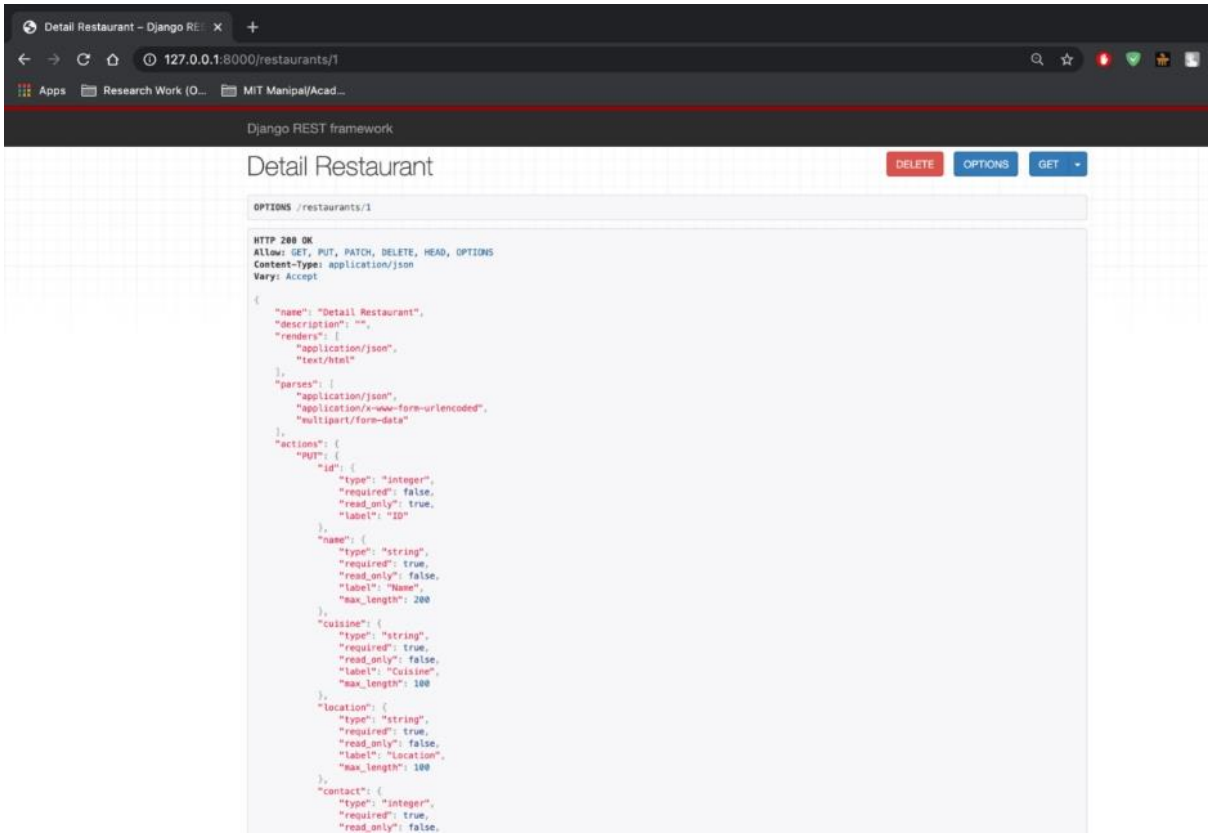

List Customer – Django REST
127.0.0.1:8000/customers
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Customer
List Customer
OPTIONS
GET
GET /customers
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"id": 1,
"name": "Tom",
"contact": 6354623457
},
{
"id": 2,
"name": "Oliver",
"contact": 7364573465
},
{
"id": 3,
"name": "Woakes",
"contact": 4478943654
}
]
Raw data
HTML form
Name
Contact
POST


Detail Customer – Django RES
127.0.0.1:8000/customers/1
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Customer / Detail Customer
Detail Customer
DELETE
OPTIONS
GET
OPTIONS /customers/1
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"name": "Detail Customer",
"description": "",
"renders": [
"application/json",
"text/html"
],
"parses": [
"application/json",
"application/x-www-form-urlencoded",
"multipart/form-data"
],
"actions": {
"PUT": {
"id": {
"type": "integer",
"required": false,
"read_only": true,
"label": "ID"
},
"name": {
"type": "string",
"required": true,
"read_only": false,
"label": "Name",
"max_length": 100
},
"contact": {
"type": "integer",
"required": true,
"read_only": false,
"label": "Contact",
"min_value": 0,
"max_value": 9223372036854775807
}
}
}
}


List Staff – Django REST frame
127.0.0.1:8000/staff
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Staff
List Staff
OPTIONS
GET
GET /staff
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"id": 1,
"name": "Chloe",
"designation": "Manager",
"contact": 7456269876
},
{
"id": 2,
"name": "Doug",
"designation": "Accountant",
"contact": 9738764569
}
]
Raw data
HTML form
Name
Designation
Contact
POST


Detail Staff – Django REST fra
127.0.0.1:8000/staff/1
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
Detail Staff
DELETE
OPTIONS
GET
OPTIONS /staff/1
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"name": "Detail Staff",
"description": "",
"renders": [
"application/json",
"text/html"
],
"parses": [
"application/json",
"application/x-www-form-urlencoded",
"multipart/form-data"
],
"actions": {
"PUT": {
"id": {
"type": "integer",
"required": false,
"read_only": true,
"label": "ID"
},
"name": {
"type": "string",
"required": true,
"read_only": false,
"label": "Name",
"max_length": 100
},
"designation": {
"type": "string",
"required": true,
"read_only": false,
"label": "Designation",
"max_length": 200
},
"contact": {
"type": "integer",
"required": true,
"read_only": false,
"label": "Contact",
"min_value": 0,
"max_value": 9223372036854775807
}
}
}
}


List Restaurant – Django REST
127.0.0.1:8000/restaurants
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Restaurant
List Restaurant
Filters
OPTIONS
GET
GET /restaurants
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[
{
"id": 1,
"name": "Egg Factory",
"cuisine": "Indian",
"location": "KC",
"contact": 7854654567
},
{
"id": 2,
"name": "BBQ Nation",
"cuisine": "Continental",
"location": "Hyderabad",
"contact": 988467356
}
]
Raw data
HTML form
Name
Cuisine
Location
Contact
POST


Detail Restaurant – Django RE
127.0.0.1:8000/restaurants/1
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
Detail Restaurant
DELETE
OPTIONS
GET
OPTIONS /restaurants/1
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"name": "Detail Restaurant",
"description": "",
"renders": [
"application/json",
"text/html"
],
"parses": [
"application/json",
"application/x-www-form-urlencoded",
"multipart/form-data"
],
"actions": {
"PUT": {
"id": {
"type": "integer",
"required": false,
"read_only": true,
"label": "ID"
},
"name": {
"type": "string",
"required": true,
"read_only": false,
"label": "Name",
"max_length": 200
},
"cuisine": {
"type": "string",
"required": true,
"read_only": false,
"label": "Cuisine",
"max_length": 100
},
"location": {
"type": "string",
"required": true,
"read_only": false,
"label": "location",
"max_length": 100
},
"contact": {
"type": "integer",
"required": true,
"read_only": false,

4. Design a ReST service for RodeSprinter, which is a one-stop solution for all local needs. Through the API, the website can request for any amenity from Fish, meat,groceries, vegetables, flowers, cakes, hotel food, home cooked food, medicines, bill payments, documents pickup and so much more. Basically, anything from anywhere.With the RodeSprinter APIs, one can search for amenities by name, or location. Identify any three Resources and implement CRUD operations.

models.py:

```
from django.db import models

# Create your models here.
class Category(models.Model):
    name = models.CharField(max_length=100)
    def __str__(self):
        return self.name

class Service(models.Model):
    name = models.CharField(max_length=200)
    provider = models.CharField(max_length=200)
    location = models.CharField(max_length=200)
    category = models.ForeignKey(Category,on_delete=models.CASCADE)
    cost = models.IntegerField()
    def __str__(self):
        return self.name

class Customer(models.Model):
```

```
    name = models.CharField(max_length=100)
    contact = models.PositiveBigIntegerField()
    def __str__(self):
        return self.name

class ServiceRequested(models.Model):
    customer = models.ForeignKey(Customer,on_delete=models.CASCADE)
    service = models.ForeignKey(Service,on_delete=models.CASCADE)
```

serializers.py:

```
from django.db.models import fields
from rest_framework import serializers
from .models import *

class CategorySerializer(serializers.ModelSerializer):
    class Meta:
        fields = '__all__'
        model = Category

class ServiceSerializer(serializers.ModelSerializer):
    class Meta:
        fields = '__all__'
        model = Service

class CustomerSerializer(serializers.ModelSerializer):
    class Meta:
        fields = '__all__'
        model = Customer

class ServiceRequestedSerializer(serializers.ModelSerializer):
    class Meta:
        fields = '__all__'
        model = ServiceRequested
```

views.py:

```
from django.shortcuts import render
from rest_framework import generics,filters
from .models import *
from .serializers import *
# Create your views here.
```

```
class ListCategory(generics.ListCreateAPIView):
    queryset = Category.objects.all()
    serializer_class = CategorySerializer

class DetailCategory(generics.RetrieveUpdateDestroyAPIView):
    queryset = Category.objects.all()
    serializer_class = CategorySerializer

class ListService(generics.ListCreateAPIView):
    queryset = Service.objects.all()
    serializer_class = ServiceSerializer
    filter_backends = [filters.SearchFilter]
    search_fields = ['name','location']

class DetailService(generics.RetrieveUpdateDestroyAPIView):
    queryset = Service.objects.all()
    serializer_class = ServiceSerializer

class ListCustomer(generics.ListCreateAPIView):
    queryset = Customer.objects.all()
    serializer_class = CustomerSerializer

class DetailCustomer(generics.RetrieveUpdateDestroyAPIView):
    queryset = Customer.objects.all()
    serializer_class = CustomerSerializer

class ListServiceRequested(generics.ListCreateAPIView):
    queryset = ServiceRequested.objects.all()
    serializer_class = ServiceRequestedSerializer

class DetailServiceRequested(generics.RetrieveUpdateDestroyAPIView):
    queryset = ServiceRequested.objects.all()
    serializer_class = ServiceRequestedSerializer
```

urls.py:

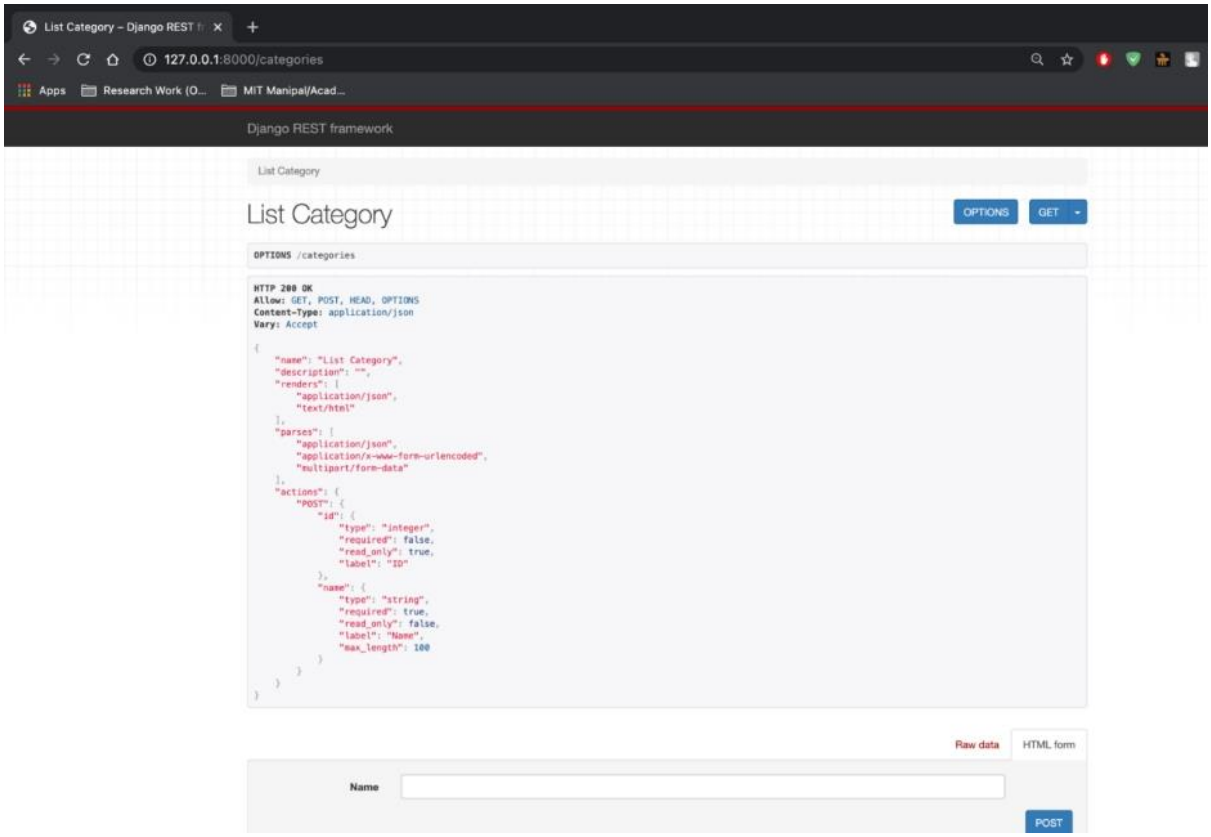
```
from django.urls import path
from .views import *

urlpatterns = [
    path('categories',ListCategory.as_view(),name="categories"),
    path('services',ListService.as_view(),name="services"),
```

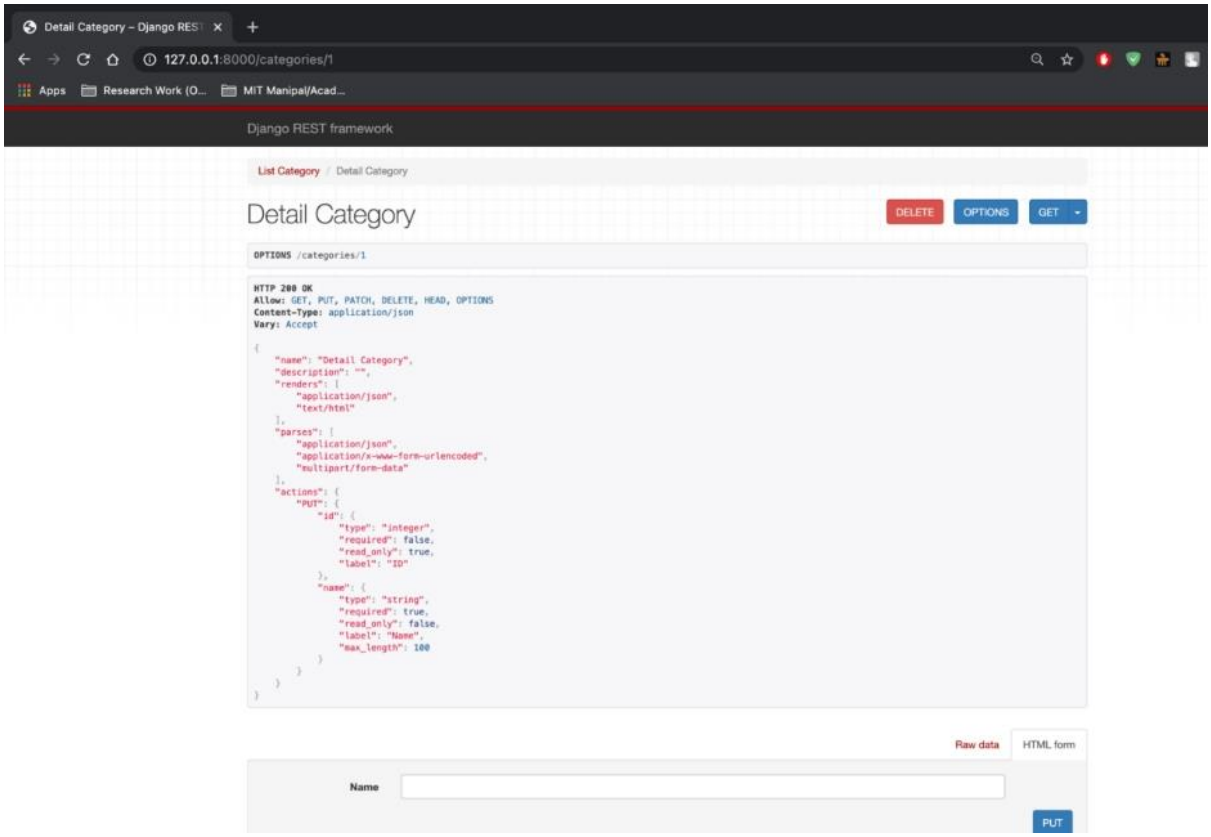
```
    path('customers',ListCustomer.as_view(),name="customers"),
    path('requests',ListServiceRequested.as_view(),name = "requests"),
    path('categories/<int:pk>',DetailCategory.as_view(),name="category"),
    path('services/<int:pk>',DetailService.as_view(),name="service"),
    path('customers/<int:pk>',DetailCustomer.as_view(),name="customer"),
    path('requests/<int:pk>',DetailServiceRequested.as_view(),name = "request"),
]
```

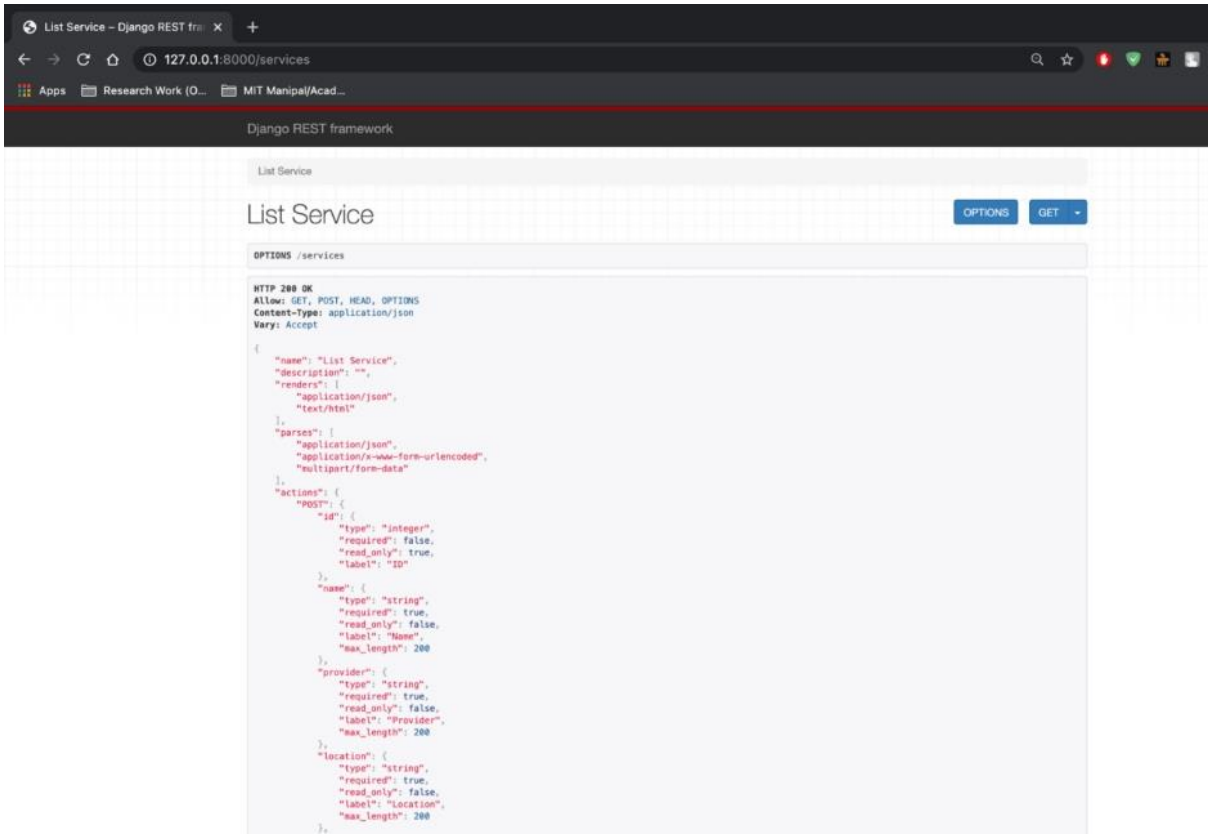
urls.py:

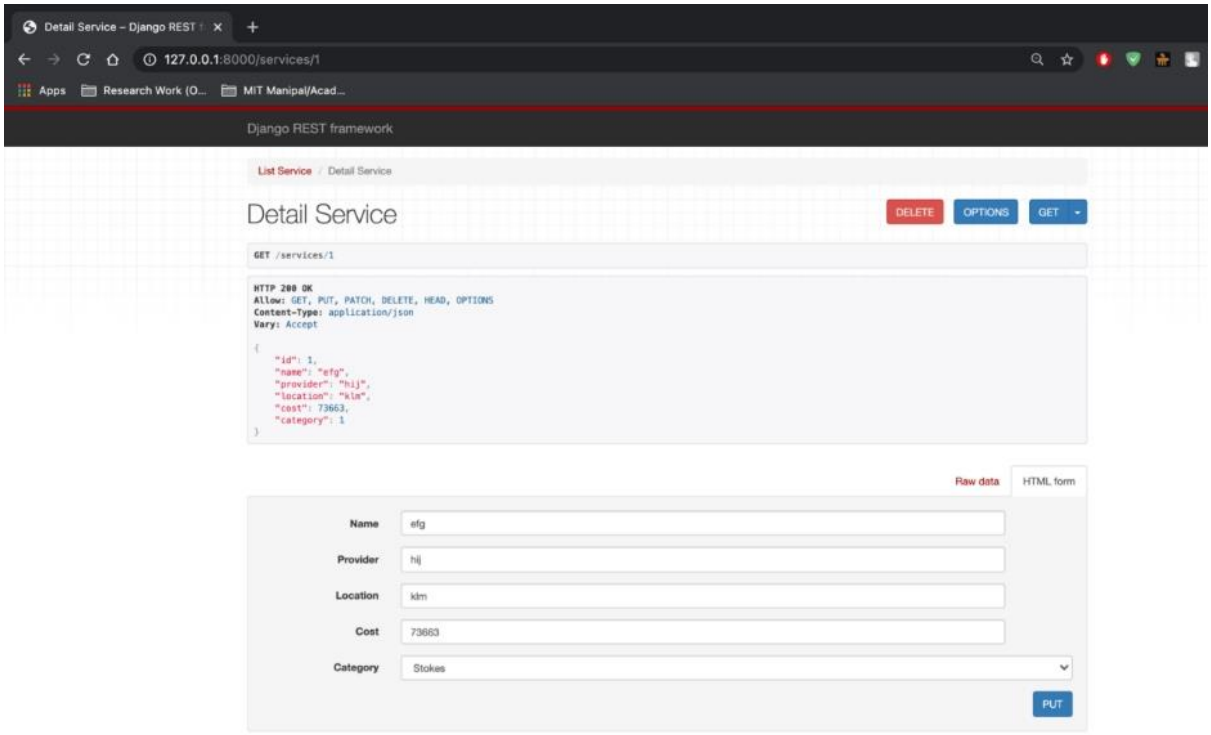
```
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    #path('',include("prob1.urls")),
    #path('',include("prob2.urls")),
    #path('',include("prob3.urls")),
    path('',include("prob4.urls")),
]
```

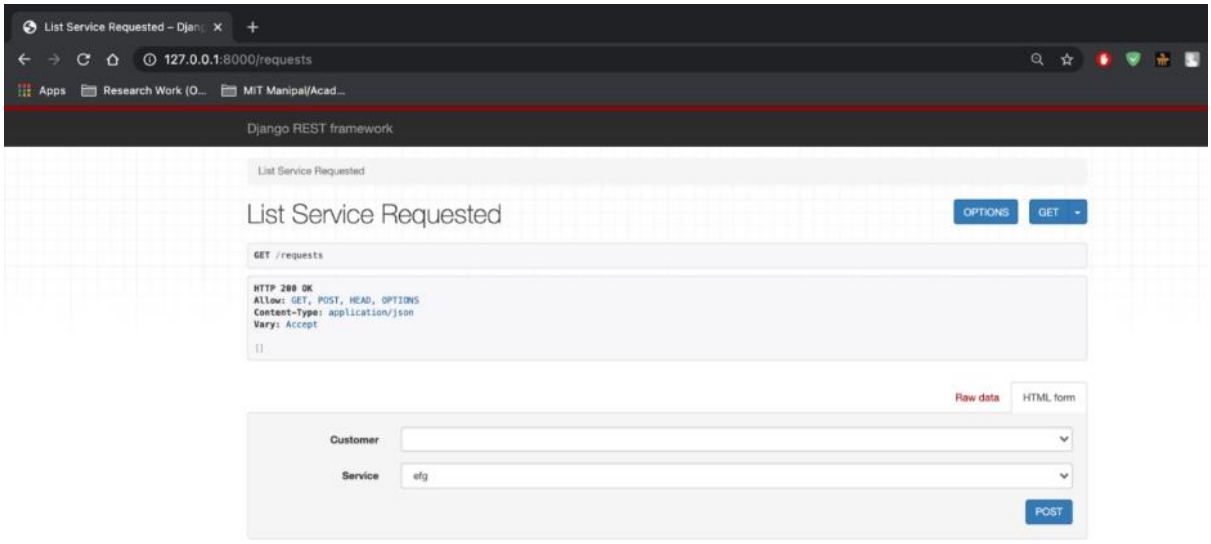
Output:

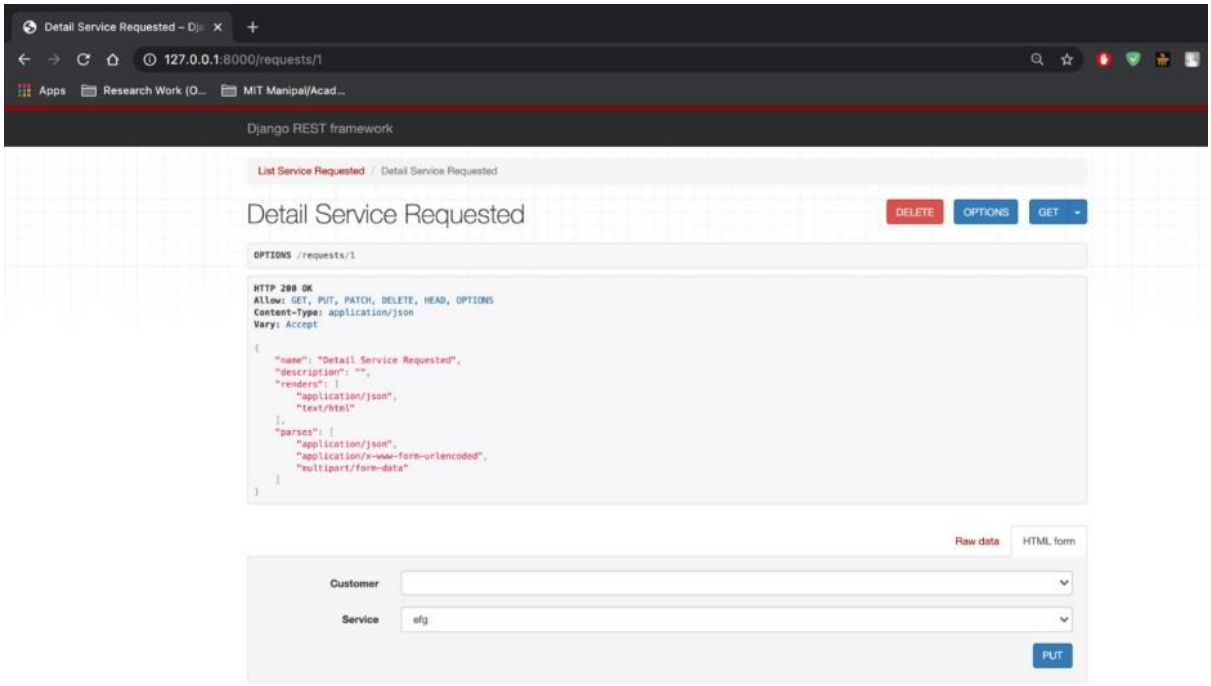

List Category – Django REST
127.0.0.1:8000/categories
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Category
List Category
OPTIONS
GET
OPTIONS /categories
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"name": "List Category",
"description": "",
"renders": [
"application/json",
"text/html"
],
"parses": [
"application/json",
"application/x-www-form-urlencoded",
"multipart/form-data"
],
"actions": {
"POST": {
"id": {
"type": "integer",
"required": false,
"read_only": true,
"label": "ID"
},
"name": {
"type": "string",
"required": true,
"read_only": false,
"label": "Name",
"max_length": 100
}
}
}
}
Raw data
HTML form
Name
POST


Detail Category – Django REST
127.0.0.1:8000/categories/1
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Category / Detail Category
Detail Category
DELETE
OPTIONS
GET
OPTIONS /categories/1
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"name": "Detail Category",
"description": "",
"renders": [
"application/json",
"text/html"
],
"parses": [
"application/json",
"application/x-www-form-urlencoded",
"multipart/form-data"
],
"actions": {
"PUT": {
"id": {
"type": "integer",
"required": false,
"read_only": true,
"label": "ID"
},
"name": {
"type": "string",
"required": true,
"read_only": false,
"label": "Name",
"max_length": 100
}
}
}
}
Raw data
HTML form
Name
PUT


List Service – Django REST fra
127.0.0.1:8000/services
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Service
List Service
OPTIONS
GET
OPTIONS /services
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"name": "List Service",
"description": "",
"renders": [
"application/json",
"text/html"
],
"parses": [
"application/json",
"application/x-www-form-urlencoded",
"multipart/form-data"
],
"actions": {
"POST": {
"id": {
"type": "integer",
"required": false,
"read_only": true,
"label": "ID"
},
"name": {
"type": "string",
"required": true,
"read_only": false,
"label": "Name",
"max_length": 200
},
"provider": {
"type": "string",
"required": true,
"read_only": false,
"label": "Provider",
"max_length": 200
},
"location": {
"type": "string",
"required": true,
"read_only": false,
"label": "Location",
"max_length": 200
},


Detail Service – Django REST
127.0.0.1:8000/services/1
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Service / Detail Service
Detail Service
DELETE
OPTIONS
GET
GET /services/1
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"id": 1,
"name": "efg",
"provider": "hij",
"location": "klm",
"cost": 73663,
"category": 1
}
Raw data
HTML form
Name
efg
Provider
hij
Location
klm
Cost
73663
Category
Stokes
PUT


List Service Requested – Djan
127.0.0.1:8000/requests
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Service Requested
List Service Requested
OPTIONS
GET
GET /requests
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
[]
Raw data
HTML form
Customer
Service
efg
POST


Detail Service Requested – Dj
127.0.0.1:8000/requests/1
Apps
Research Work (O...
MIT Manipal/Acad...
Django REST framework
List Service Requested / Detail Service Requested
Detail Service Requested
DELETE
OPTIONS
GET
OPTIONS /requests/1
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
{
"name": "Detail Service Requested",
"description": "",
"renders": [
"application/json",
"text/html"
],
"parses": [
"application/json",
"application/x-www-form-urlencoded",
"multipart/form-data"
]
}
Raw data
HTML form
Customer
Service
efg
PUT