IT LAB : WEEK 10 REST API

NAME:SAGNIK CHATTERJEE

REG: 180905478

ROLL NO : 61 SEC :B

Q1. 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.

Project level :

Settings.py

```
"""
Django settings for q1 project.

Generated by 'django-admin startproject' using Django 3.2.3.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""
import os
from pathlib import Path

# 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-+c#%xf5k1u++vi#c8%3&3cteq_!e%q34u#gwv_)y)w=sc1&
hm5'

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

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    "rest_framework",
    "blog",
]
```

```
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 = 'q1.urls'

TEMPLATES = [
    {
        'BACKEND':
'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        '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 = 'q1.wsgi.application'


# Database
```

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

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


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

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarity
Validator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator'
,
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator
',
    },
    {
```

```
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidato
r',
    },
]


# 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'
```

Urls.py

```
"""q1 URL Configuration

The `urlpatterns` list routes URLs to views. For more
information please see:
    https://docs.djangoproject.com/en/3.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home,
name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(),
name='home')
Including another URLconf
    1. Import the include() function: from django.urls import
include, path
    2. Add a URL to urlpatterns:  path('blog/',
include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path , include

urlpatterns = [
    path('admin/', admin.site.urls),
    path("blog/",include("blog.urls")),
]
```

App level:

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"),
]
```

Serializers.py

```
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
```

Models.py

```
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 + " " + self.email


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 + " " + self.date


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()

    def __str__(self):
        return self.user + " " + self.comment
```

Views.py

```
from django.shortcuts import render
from django.http import request

# Create your views here.
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
```

Django REST framework sagnik

List User

List User

OPTIONS GET

```
POST /blog/users
```

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 2,
    "username": "lambda2",
    "email": "lambda1.2@gmail.com",
    "phno": 234567890,
    "password": "qwerty"
}
```

Raw data HTML form

Username lambda2

Email lambda1.2@gmail.com

Phno 234567890

Password qwerty

POST

http://127.0.0.1:8000/blog/users

Save

GET http://127.0.0.1:8000/blog/users Send

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

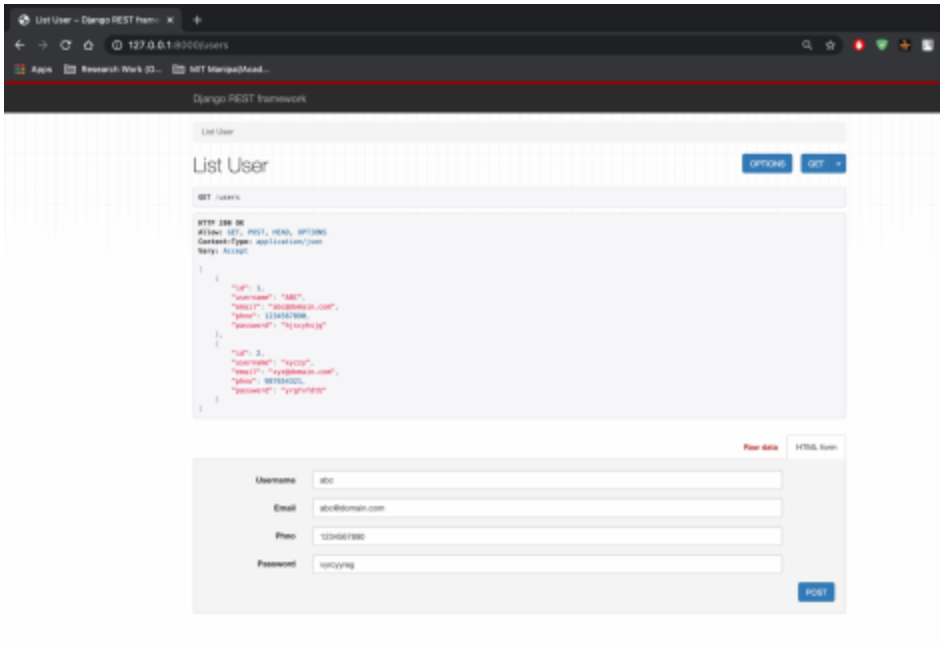
Query Params

KEY	VALUE	DESCRIPTION	... Bulk Edit
Key	Value	Description	

Body Cookies Headers (9) Test Results Status: 200 OK Time: 17 ms Size: 480 B Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "username": "lambda1",
        "email": "lambda@gmail.com",
        "phno": 123456789,
        "password": "asigrf"
    },
    {
        "id": 2,
        "username": "lambda2",
        "email": "lambda1.2@gmail.com",
        "phno": 234567890,
        "password": "qwerty"
    }
]
```



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.

Project level:

Settings.py

```
"""
Django settings for q2 project.

Generated by 'django-admin startproject' using Django 3.2.3.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/
```

```
For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""
import os
from pathlib import Path

# 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-c8es!gi@)8dkm3no4a8@ju61h^t9m7#xtb*&9je6n#yj(k-
ei2"

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

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
```

```
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
    "rest_framework",
    "ola",
]

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 = "q2.urls"

TEMPLATES = [
    {
        "BACKEND":
"django.template.backends.django.DjangoTemplates",
        "DIRS": [],
        "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 = "q2.wsgi.application"


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

DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": BASE_DIR / "db.sqlite3",
    }
}


# 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.NumericPasswordValidato
r",
    },
]


# 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"
```

Urls.py

```
"""q2 URL Configuration

The `urlpatterns` list routes URLs to views. For more
information please see:
    https://docs.djangoproject.com/en/3.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home,
name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(),
name='home')
Including another URLconf
    1. Import the include() function: from django.urls import
include, path
    2. Add a URL to urlpatterns:  path('blog/',
include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path("admin/", admin.site.urls),
    path("ola/", include("ola.urls")),
]
```

App level:
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()

    def __str__(self):
        return self.userLocation + " " + self.vehicle
```

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
```

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"),
]
```

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
```

```python
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
```

Django REST framework sagnik

List User Data

List User Data

OPTIONS GET

POST /ola/userData

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 2,
    "name": "chatterjee2",
    "contact": 123456796
}
```

Raw data HTML form

Name chatterjee2

Contact 123456796

POST

http://127.0.0.1:8000/ola/userData

Save

GET http://127.0.0.1:8000/ola/userData Send

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

Query Params

KEY	VALUE	DESCRIPTION	... Bulk Ed
Key	Value	Description	

Body Cookies Headers (9) Test Results Status: 200 OK Time: 30 ms Size: 389 B Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "name": "chatterjee1",
        "contact": 123456789
    },
    {
        "id": 2,
        "name": "chatterjee2",
        "contact": 123456796
    }
]
```

Django REST framework sagnik

List Vehicle Info

List Vehicle Info

OPTIONS GET

POST /ola/vehicleInfo

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "driverName": "chintu",
    "vehicleName": "verna",
    "vehicleRegNo": "KA12J4567",
    "contact": 91231312313213
}
```

Raw data HTML form

DriverName	chintu
VehicleName	verna
VehicleRegNo	KA12J4567
Contact	91231312313213

POST

http://127.0.0.1:8000/ola/userLocation

Save

GET http://127.0.0.1:8000/ola/userLocation Send

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Bulk Ed

Body Cookies Headers (9) Test Results Status: 200 OK Time: 24 ms Size: 351 B Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "latitude": "123.4567",
        "longitude": "345.6789",
        "user": 1
    }
]
```

Django REST framework sagnik

List User Location

List User Location

OPTIONS GET

```
POST /ola/userLocation
```

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "latitude": "123.4567",
    "longitude": "345.6789",
    "user": 1
}
```

Raw data HTML form

Latitude 123.4567

Longitude 345.6789

User chatterjee1

POST

http://127.0.0.1:8000/ola/vehicleInfo

Save

GET http://127.0.0.1:8000/ola/vehicleInfo Send

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

Query Params

KEY	VALUE	DESCRIPTION	••• Bulk Ed
Key	Value	Description	

Body Cookies Headers (9) Test Results Status: 200 OK Time: 22 ms Size: 394 B Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "driverName": "chintu",
        "vehicleName": "verna",
        "vehicleRegNo": "KA12J4567",
        "contact": 91231312313213
    }
]
```

Django REST framework sagnik

List Travel Status

List Travel Status

OPTIONS GET

```
POST /ola/travelStatus
```

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 2,
    "eta": "14:21:00",
    "fare": 123467,
    "userLocation": 1,
    "vehicle": 1
}
```

Raw data HTML form

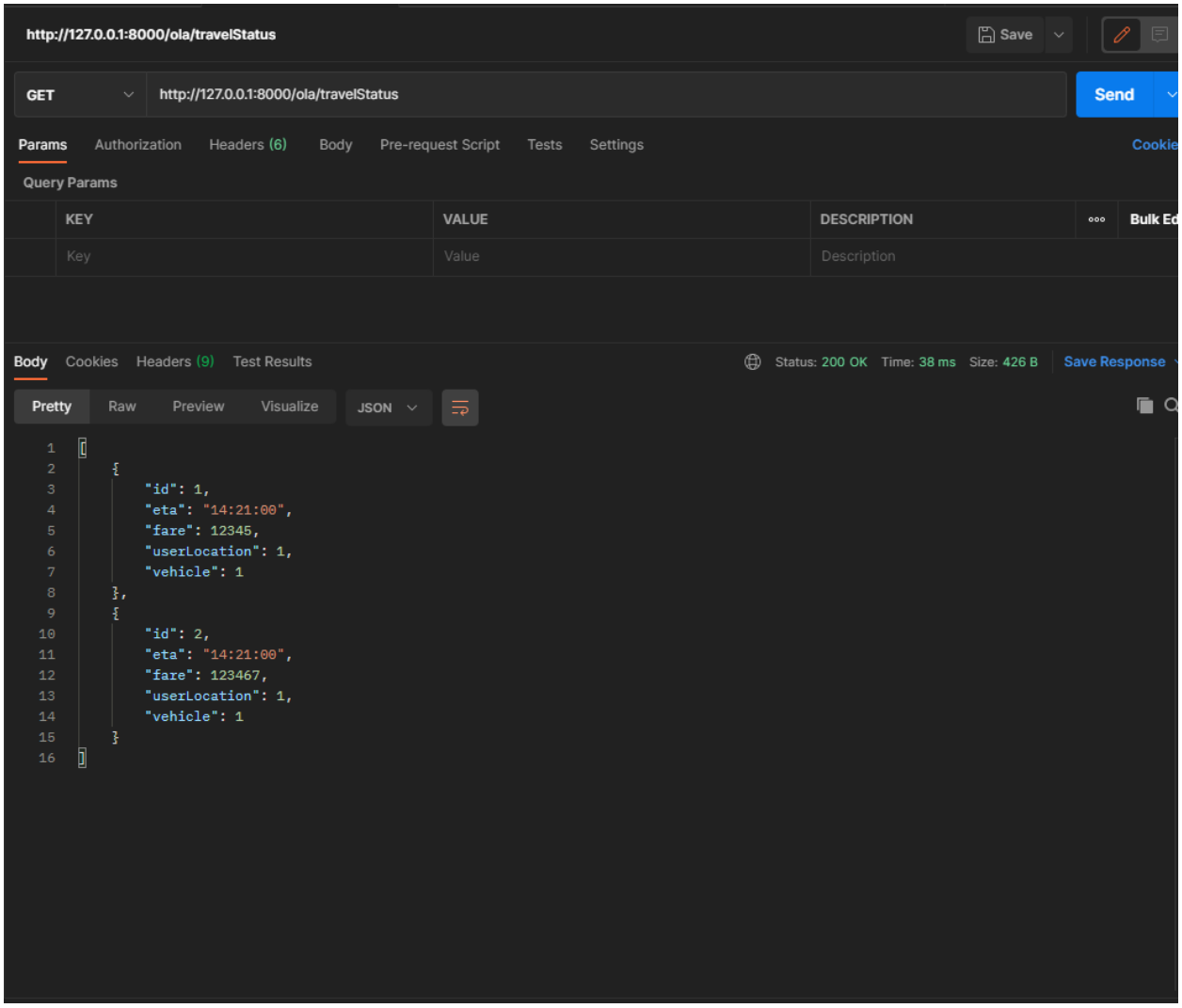
Eta 14:21

Fare 123467

UserLocation 123.4567_345.6789_chatterjee1

Vehicle chintu_verna_KA12J4567

POST



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, or location. Identify any three resources and implement CRUD operations.

Project level :
Settings.py

```
"""
Django settings for q3 project.
```

```
Generated by 'django-admin startproject' using Django 3.2.3.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""
import os
from pathlib import Path

# 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-d5u@)#g(2dlbxn4-k88or$n+^q2yhgub*i=-r&dym)na56e
u*6'

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

ALLOWED_HOSTS = []


# Application definition
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    "rest_framework",
    "zomato",
]

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 = 'q3.urls'

TEMPLATES = [
    {
        'BACKEND':
'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        '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 = 'q3.wsgi.application'


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

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


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

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarity
Validator',
    },
    {
```

```
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator'
,
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator
',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidato
r',
    },
]


# 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'
```

Urls.py

```
"""q3 URL Configuration

The `urlpatterns` list routes URLs to views. For more
information please see:
    https://docs.djangoproject.com/en/3.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home,
name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(),
name='home')
Including another URLconf
    1. Import the include() function: from django.urls import
include, path
    2. Add a URL to urlpatterns:  path('blog/',
include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
```

```
    path("admin/", admin.site.urls),
    path("zomato/", include("zomato.urls")),
]
```

App level:
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
```

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"),
]
```

Views.py

```python
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
```

Overview | GET http://127.0.0.1:80... | No Environment

http://127.0.0.1:8000/zomato/customers

Save

GET http://127.0.0.1:8000/zomato/customers Send

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body | Cookies | Headers (9) | Test Results | Status: 200 OK | Time: 41 ms | Size: 389 B | Save Response

Pretty | Raw | Preview | Visualize | JSON


```
[
    {
        "id": 1,
        "name": "customer1",
        "contact": 12313123123123
    },
    {
        "id": 2,
        "name": "customer2",
        "contact": 12313123
    }
]
```

Django REST framework sagnik

List Restaurant

List Restaurant

OPTIONS GET

POST /zomato/restaurants

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "name": "resturant 2",
    "cuisine": "hakka noodles",
    "location": "chinatown.delhi",
    "contact": 667788995
}
```

Raw data HTML form

Name	resturant 2
Cuisine	hakka noodles
Location	chinatown.delhi
Contact	667788995

POST

http://127.0.0.1:8000/zomato/staff

Save

GET http://127.0.0.1:8000/zomato/staff Send

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Bulk Ed

Body Cookies Headers (9) Test Results

Status: 200 OK Time: 35 ms Size: 367 B Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "name": "chintu",
        "designation": "head chef",
        "contact": 12313123123123123
    }
]
```

Django REST framework sagnik

List Staff

List Staff

OPTIONS GET

```
POST /zomato/staff
```

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "name": "chintu",
    "designation": "head chef",
    "contact": 12313123123123123
}
```

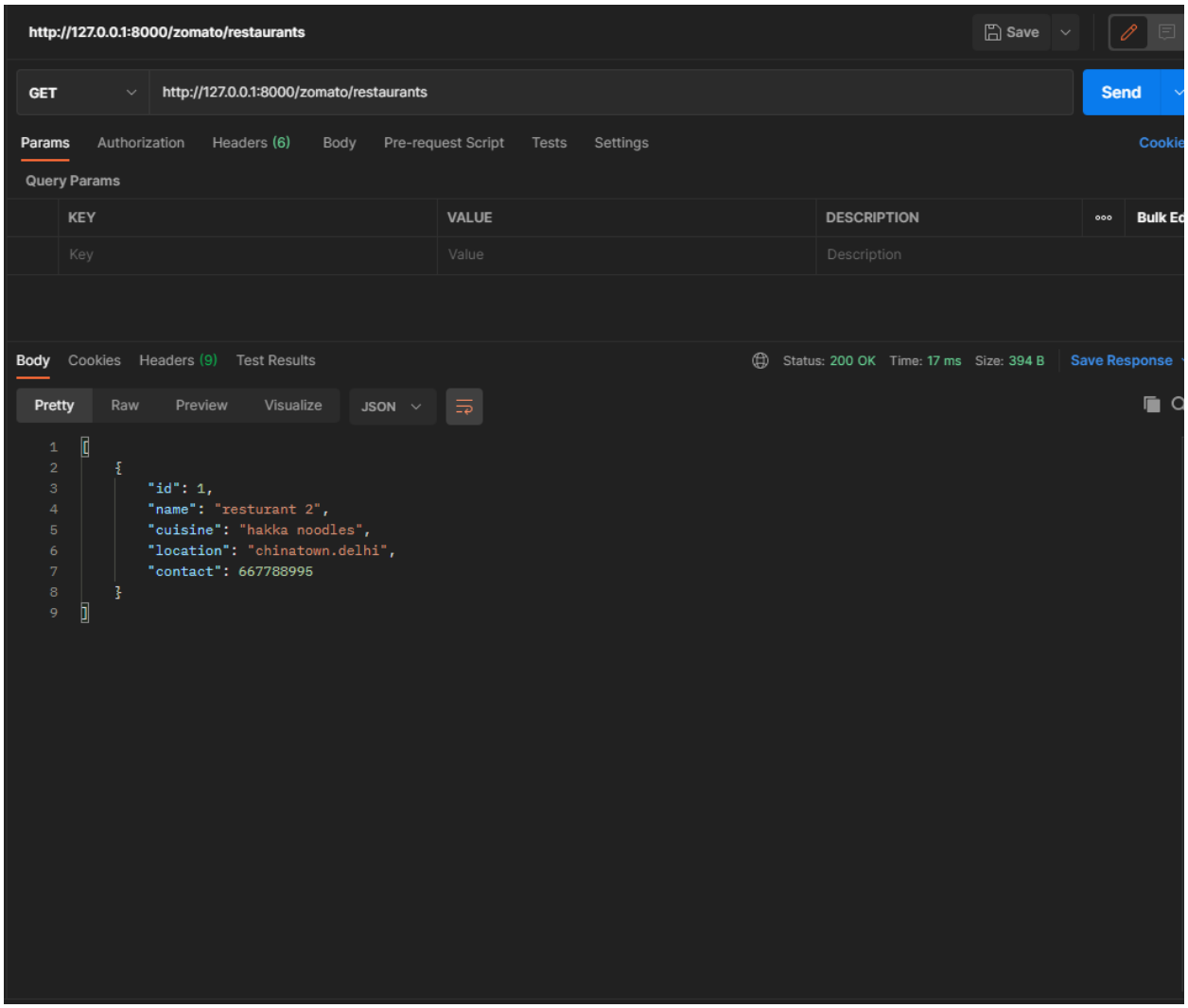
Raw data HTML form

Name chintu

Designation head chef

Contact 12313123123123123

POST



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.

Project level

Settings.py

```
"""
Django settings for q4 project.

Generated by 'django-admin startproject' using Django 3.2.3.

For more information on this file, see
https://docs.djangoproject.com/en/3.2/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/3.2/ref/settings/
"""
import os
from pathlib import Path

# 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-claxovgj6kzo^$ynhj0o#=!s06!6(vsje*g(tlbl(z*$n%o
5pl'

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

ALLOWED_HOSTS = []
```

```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    "rest_framework",
    "amazon",
]

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 = 'q4.urls'

TEMPLATES = [
    {
        'BACKEND':
'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        '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 = 'q4.wsgi.application'


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

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}


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

AUTH_PASSWORD_VALIDATORS = [
    {
```

```
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarity
Validator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator'
,
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator
',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidato
r',
    },
]


# 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'
```

Urls.py

```
"""q4 URL Configuration

The `urlpatterns` list routes URLs to views. For more
information please see:
    https://docs.djangoproject.com/en/3.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home,
name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(),
name='home')
Including another URLconf
    1. Import the include() function: from django.urls import
include, path
    2. Add a URL to urlpatterns:  path('blog/',
include('blog.urls'))
```

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

urlpatterns = [
    path("admin/", admin.site.urls),
    path("amazon/", include("amazon.urls")),
]
```

App level:
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)

    def __str__(self):
        return self.customer + "" + self.service
```

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
```

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"),
]
```

Views.py

```
from django.shortcuts import render
```

```python
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
```

Django REST framework sagnik

List Category

List Category

OPTIONS GET

```
GET /amazon/categories
```

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 1,
        "name": "some category 1"
    },
    {
        "id": 2,
        "name": "List Category"
    }
]
```

Raw data HTML form

Name

POST

http://127.0.0.1:8000/amazon/categories

Save

GET http://127.0.0.1:8000/amazon/categories Send

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

Query Params

KEY	VALUE	DESCRIPTION	••• Bulk Ed
Key	Value	Description	

Body Cookies Headers (9) Test Results Status: 200 OK Time: 25 ms Size: 354 B Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "name": "some category 1"
    },
    {
        "id": 2,
        "name": "List Category"
    }
]
```

Django REST framework sagnik

List Service

List Service

OPTIONS GET

POST /amazon/services

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 2,
    "name": "List Service",
    "provider": "RiderProvider",
    "location": "Lahore",
    "cost": 1234,
    "category": 2
}
```

Raw data HTML form

Name	List Service
Provider	RiderProvider
Location	Lahore
Cost	1234
Category	List Category

POST

http://127.0.0.1:8000/amazon/services

Save

GET http://127.0.0.1:8000/amazon/services Send

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

Query Params

KEY	VALUE	DESCRIPTION	••• Bulk Ed
Key	Value	Description	

Body Cookies Headers (9) Test Results Status: 200 OK Time: 12 ms Size: 506 B Save Response

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": 1,
        "name": "service provider 1",
        "provider": "provider 1",
        "location": "japan",
        "cost": 1231312312313,
        "category": 1
    },
    {
        "id": 2,
        "name": "List Service",
        "provider": "RiderProvider",
        "location": "Lahore",
        "cost": 1234,
        "category": 2
    }
]
```

Django REST framework sagnik

List Service Requested

List Service Requested

OPTIONS GET

```
GET /amazon/requests
```

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[]
```

Raw data HTML form

Customer

Service List Service

POST