**IT Lab  9: REST API**

**Name:** GP Anirudh

**Roll Number:** 59

**Section:** B

**Batch:** B2

**Registration Number:** 180905452 

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

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:**

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")),

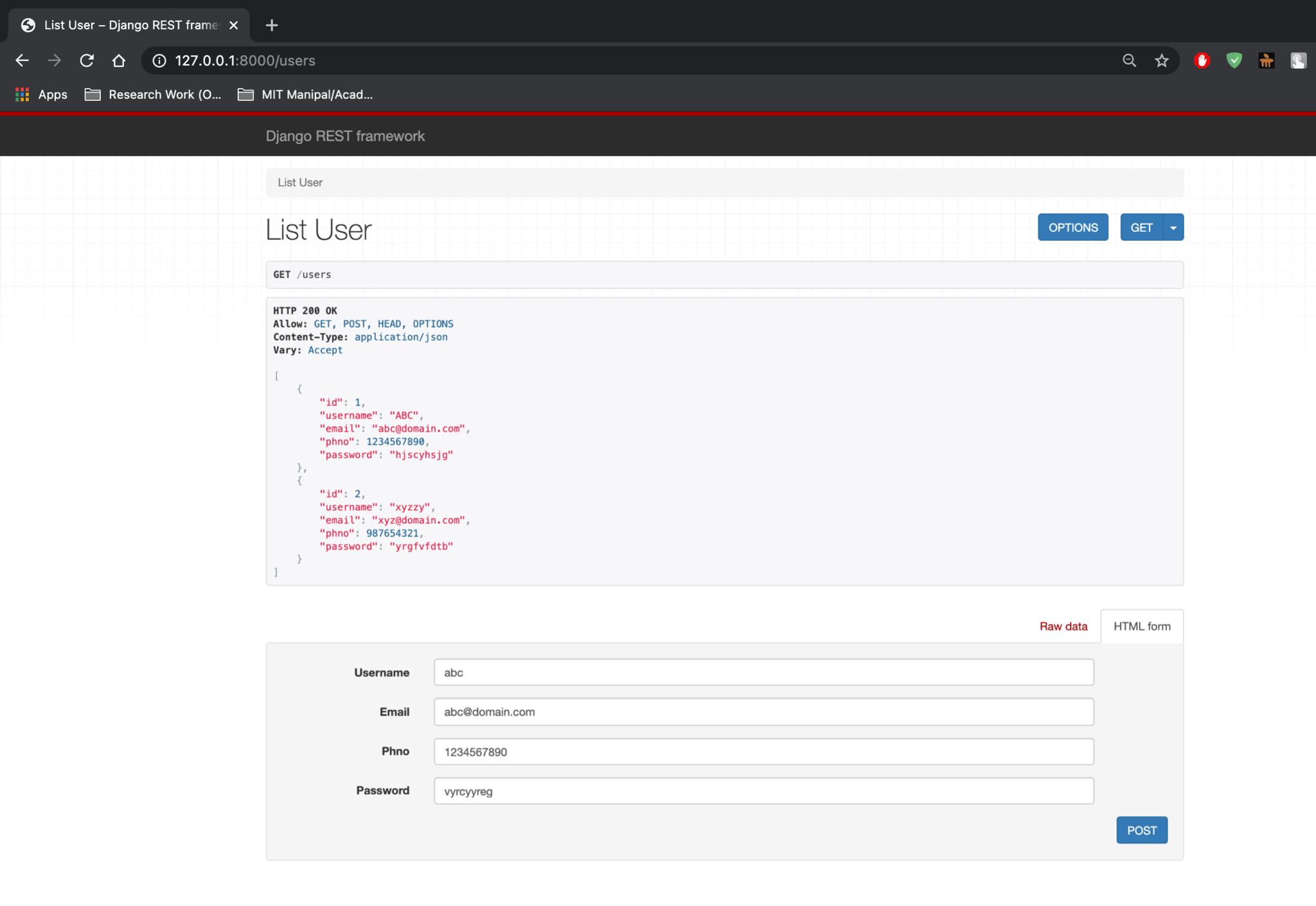
#path('',include("prob2.urls")),

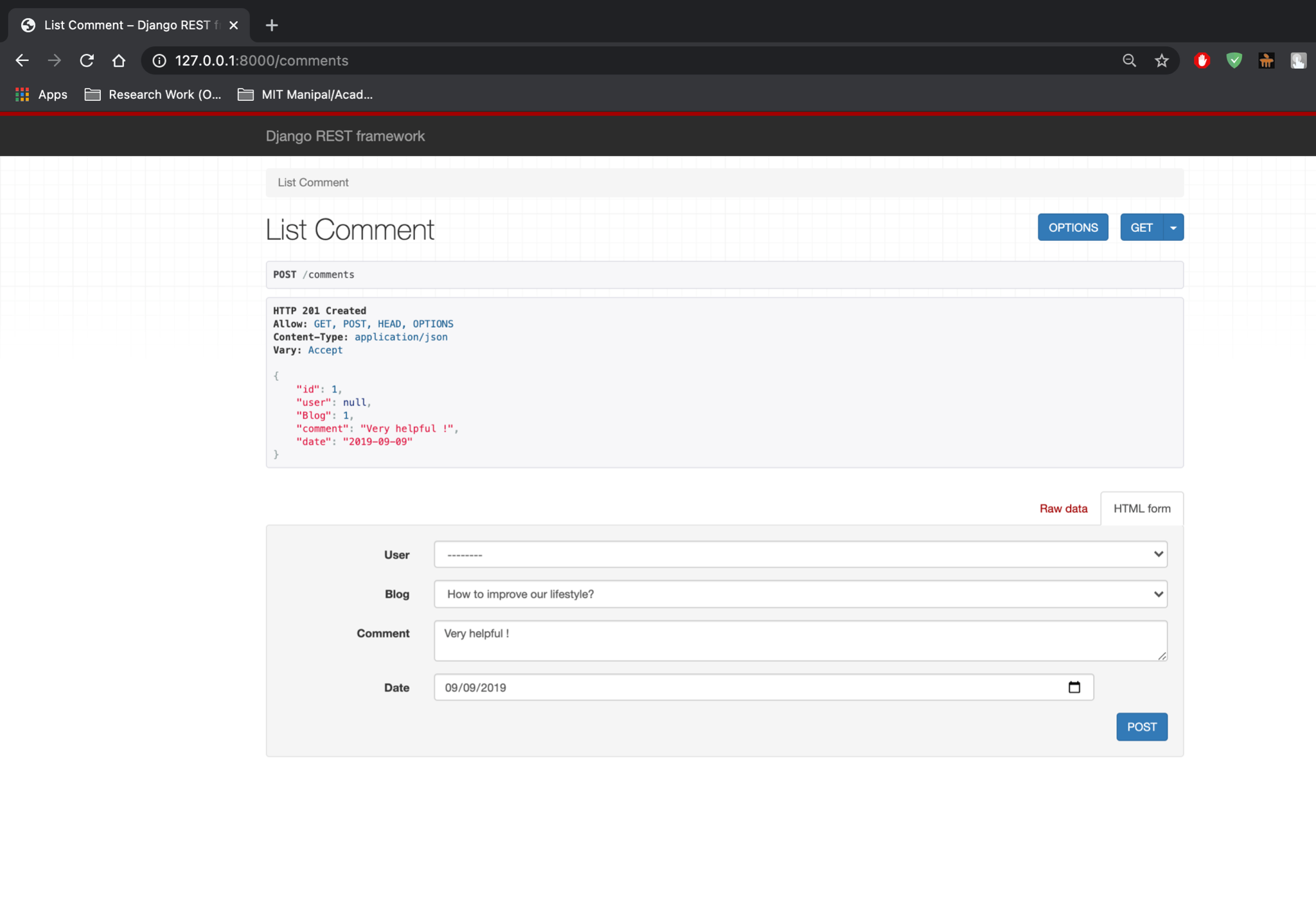
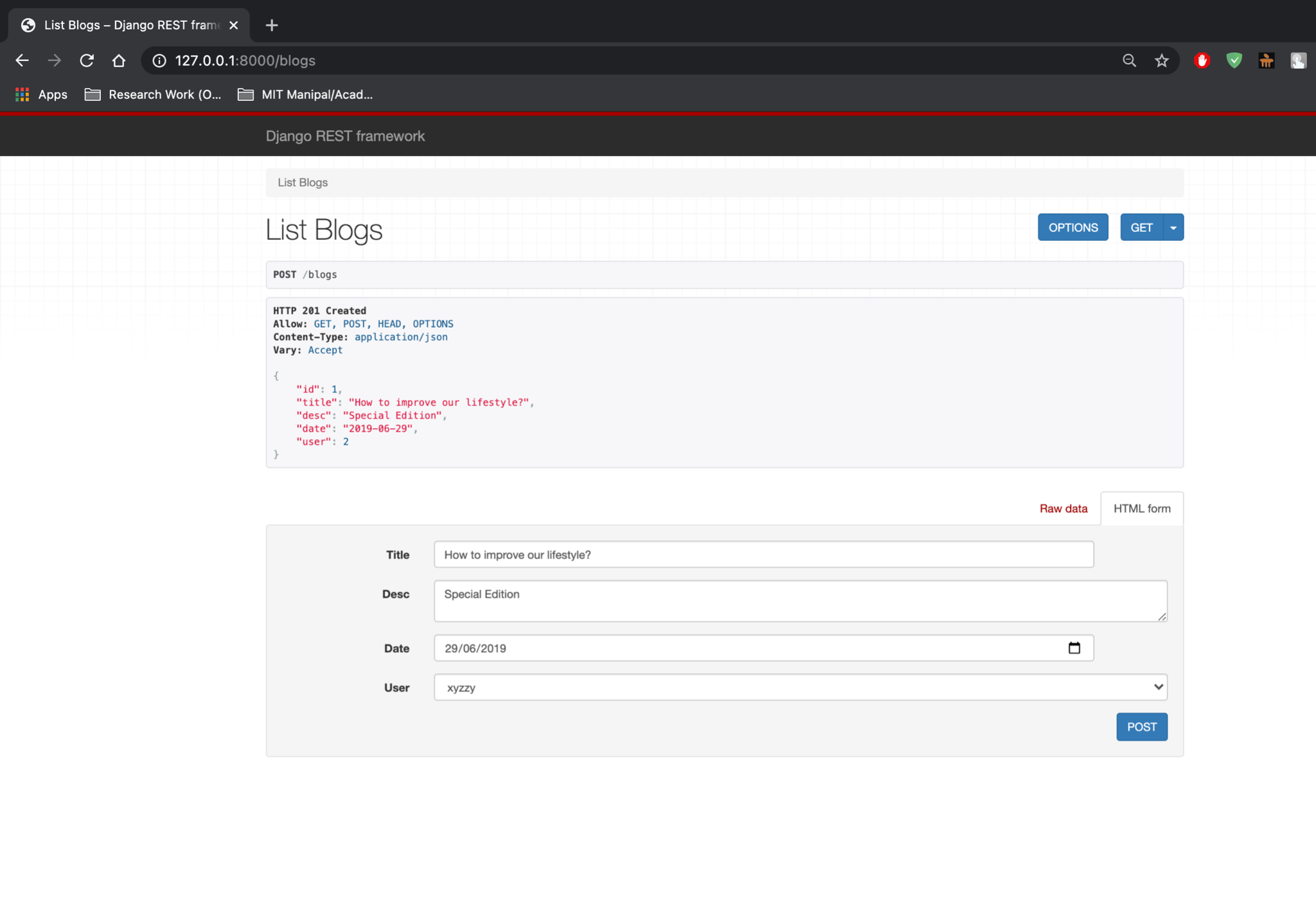
#path('',include("prob3.urls")),

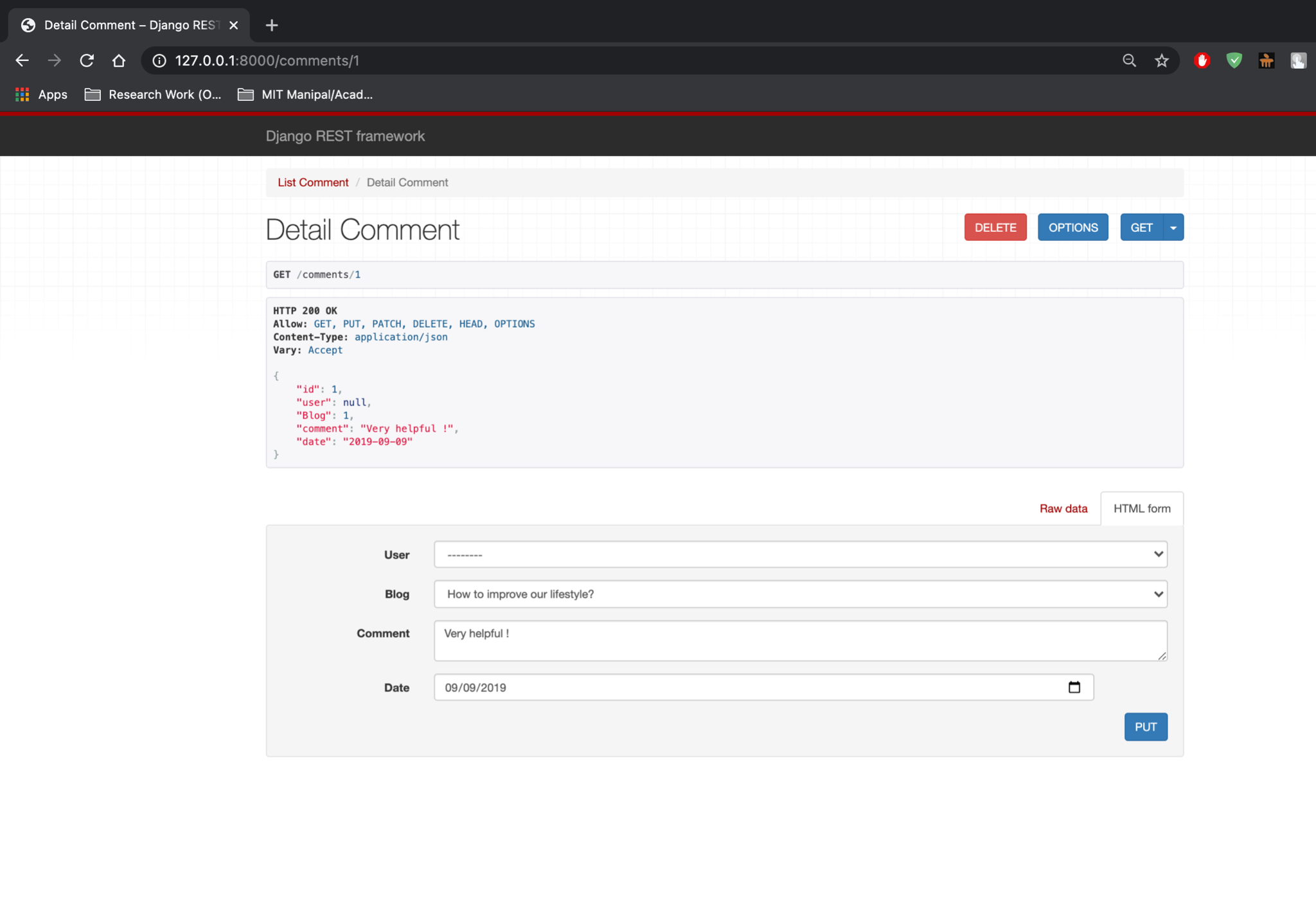
#path('',include("prob4.urls")),

]

**Output:**

****

****

****

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")),

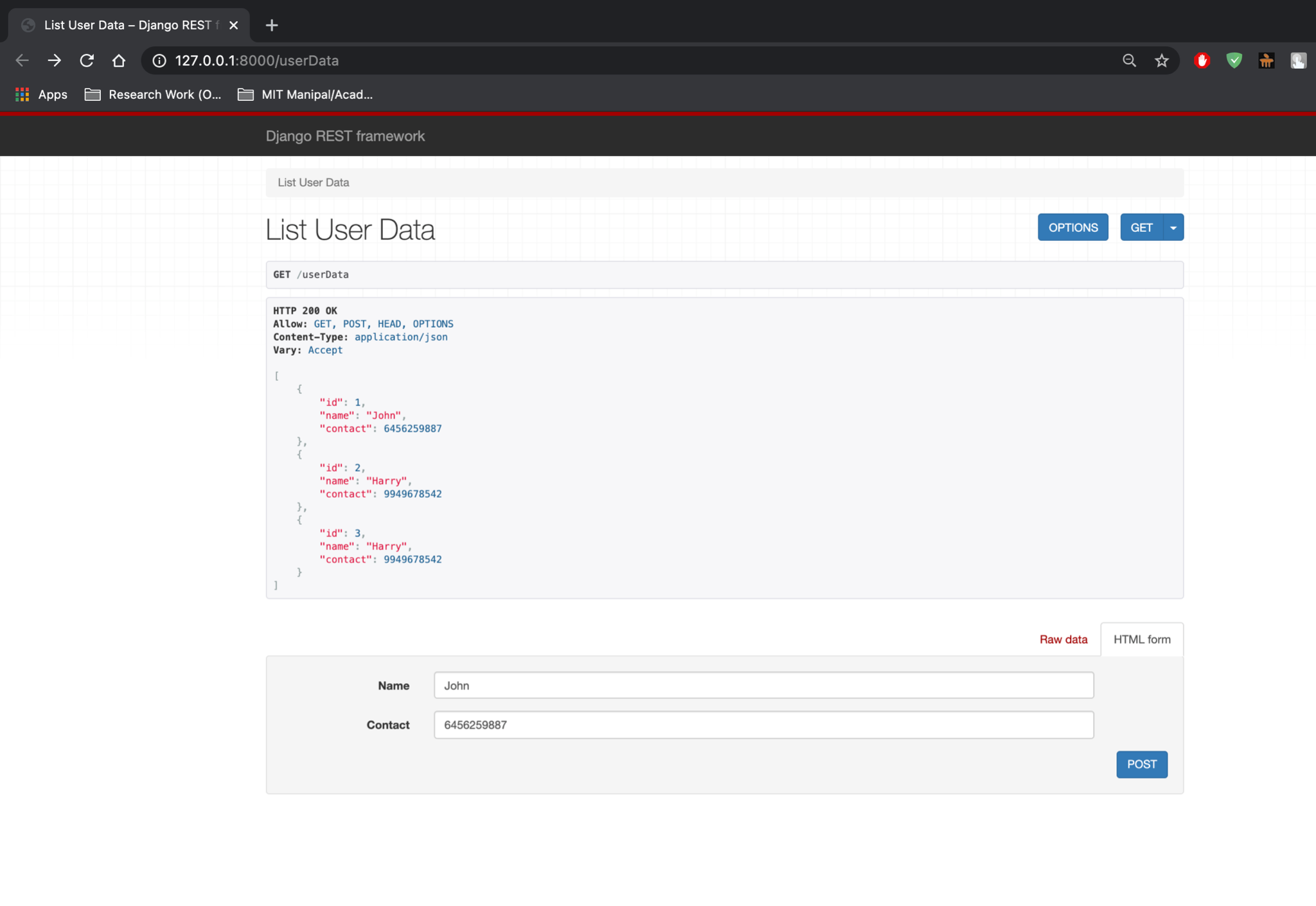
path('',include("prob2.urls")),

#path('',include("prob3.urls")),

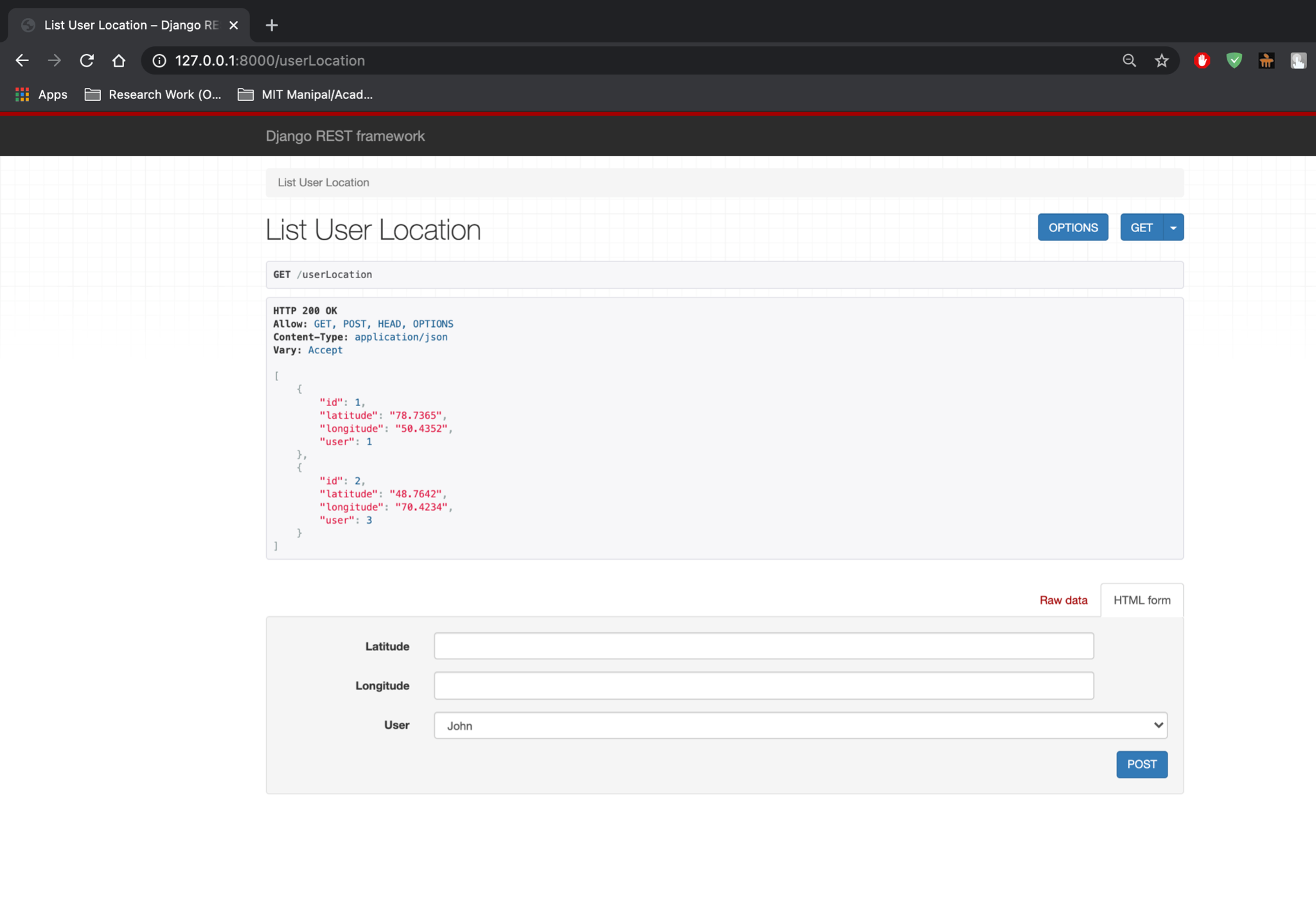
#path('',include("prob4.urls")),

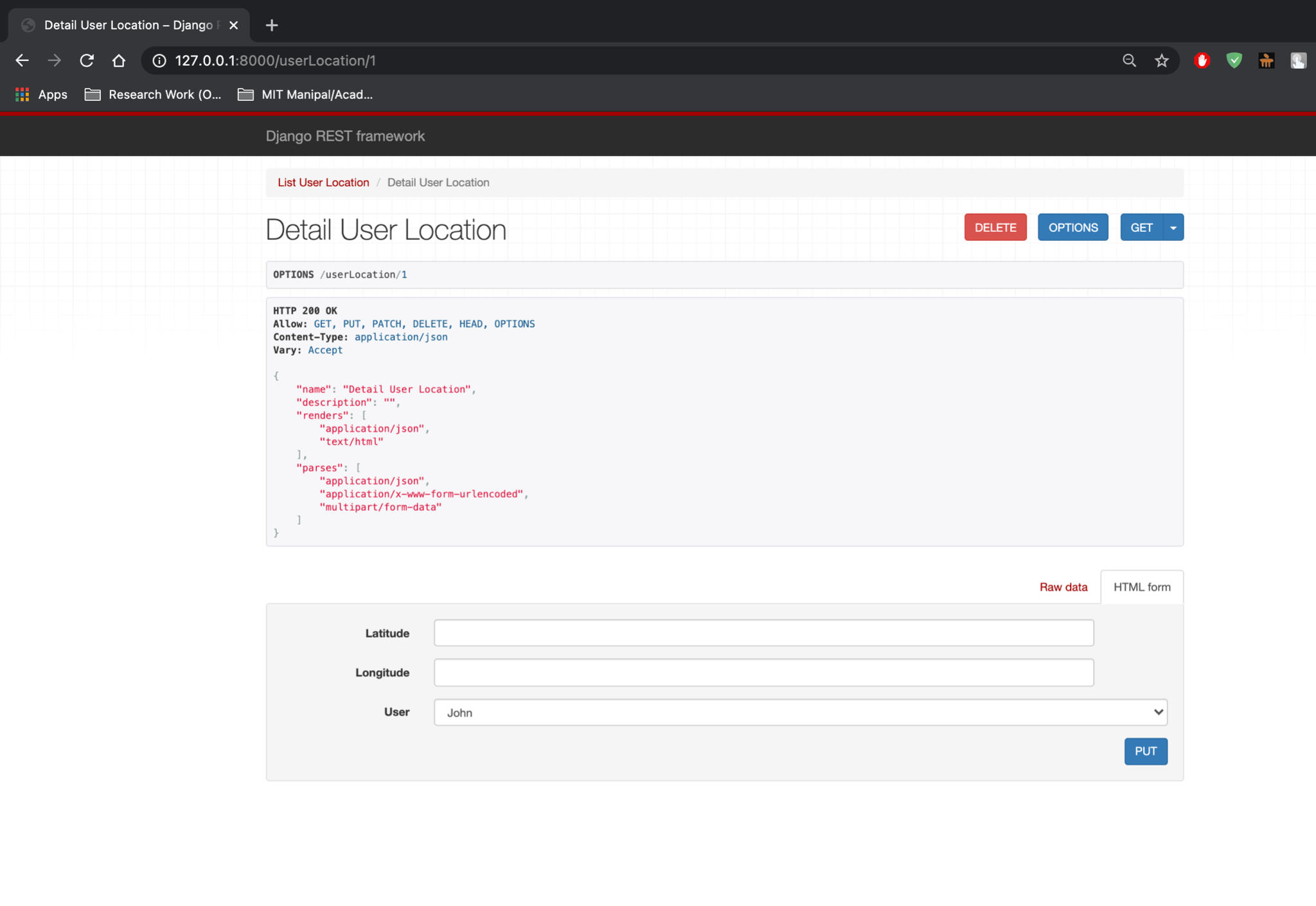
]

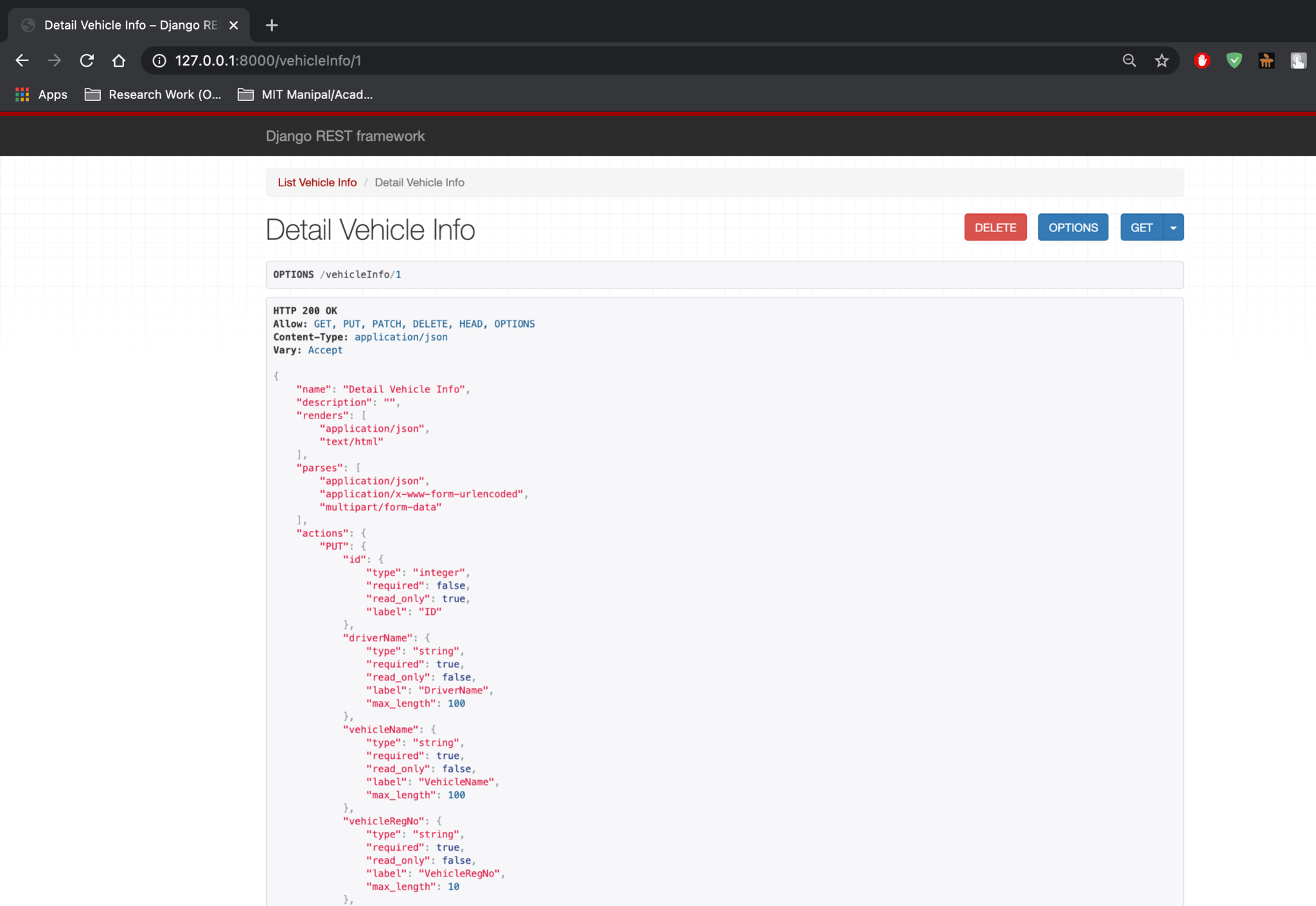
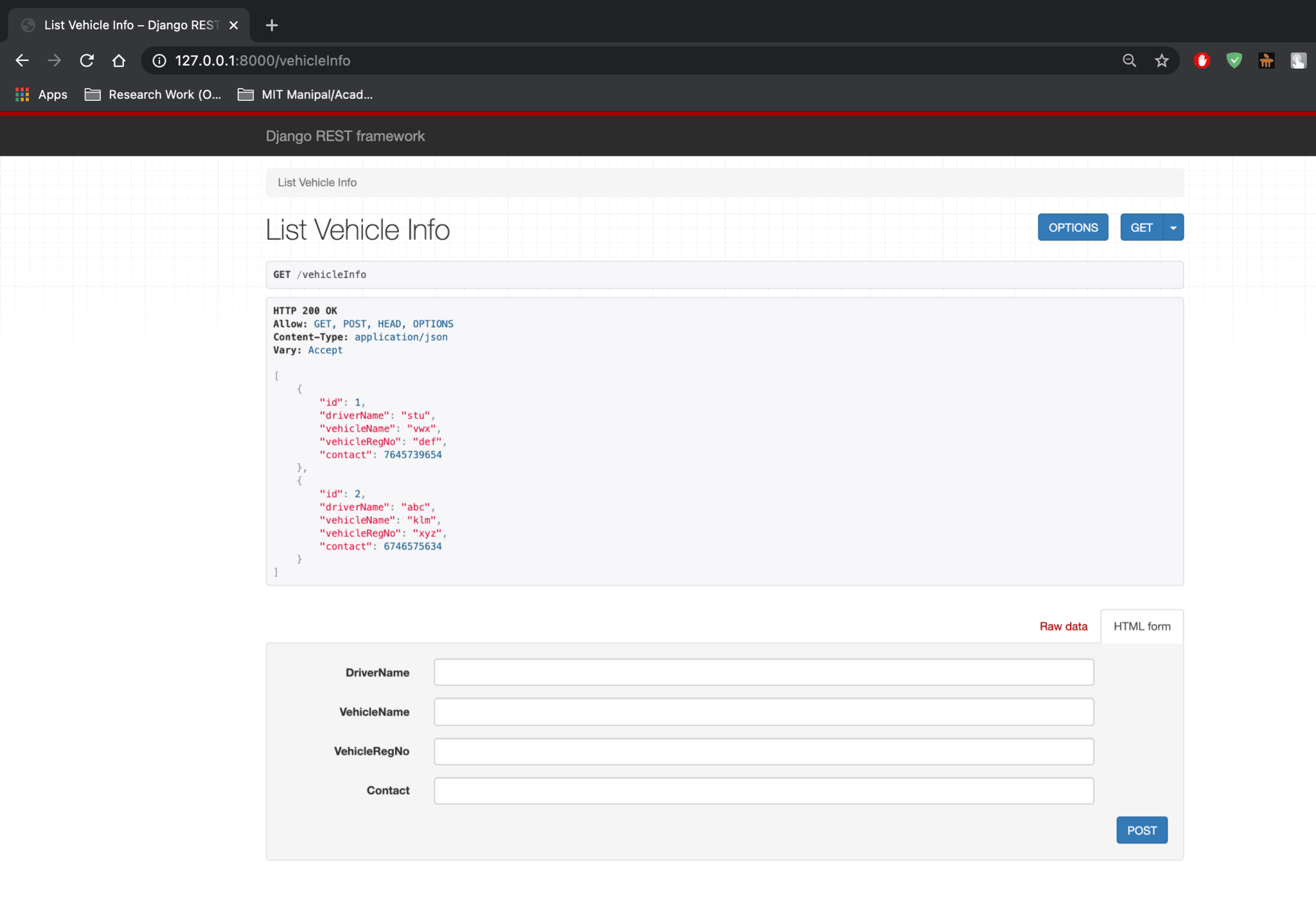
**Output:**

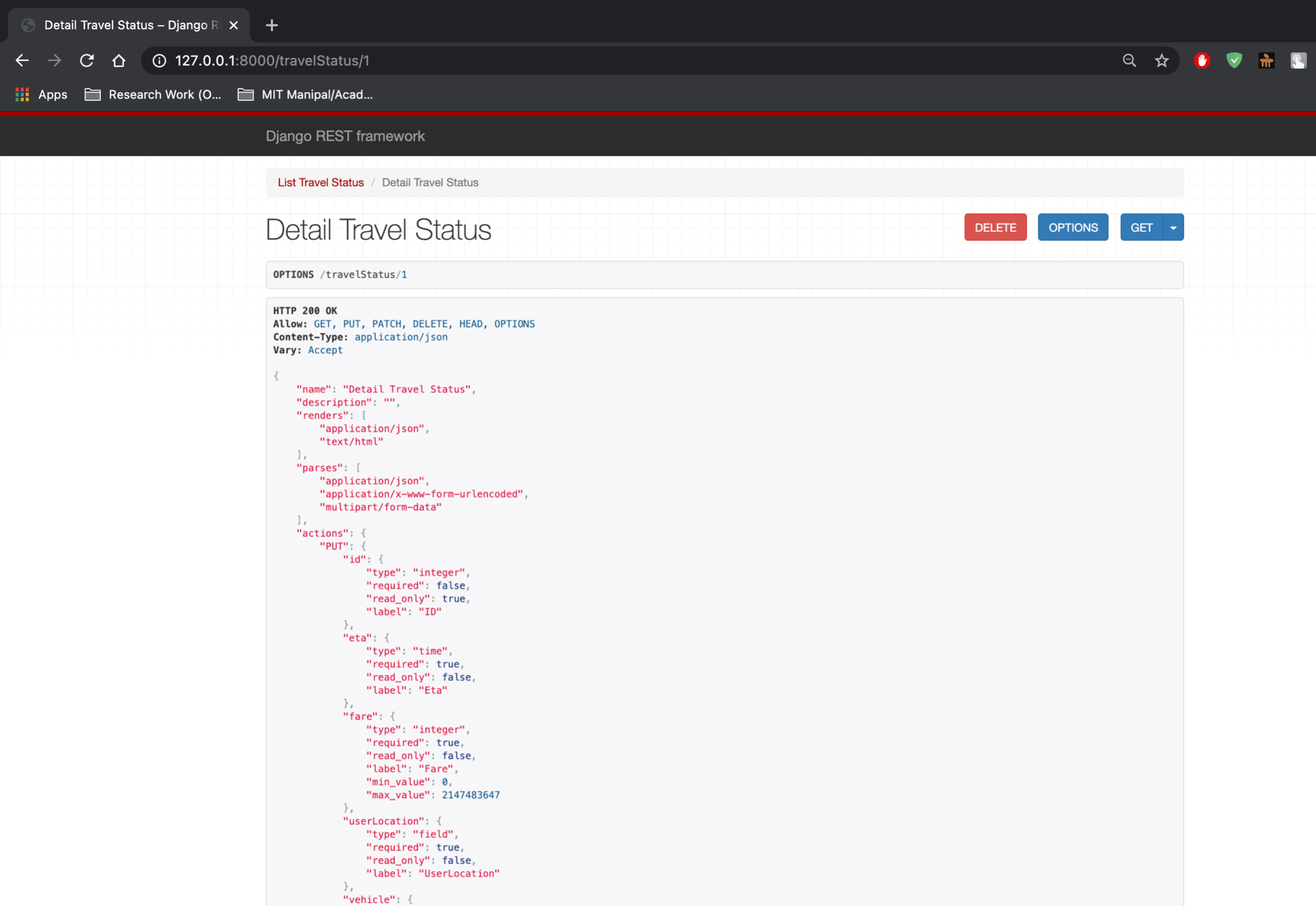
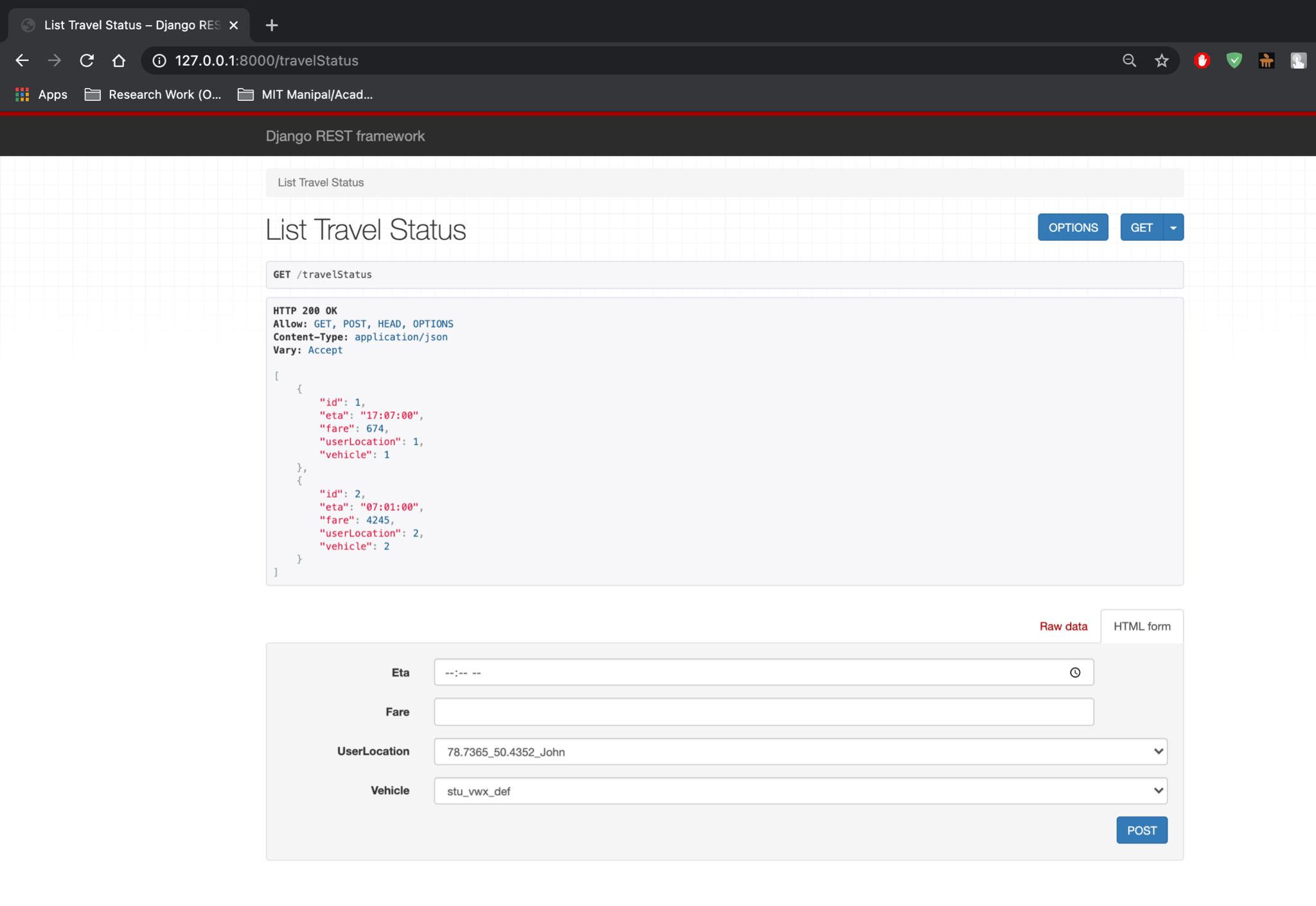
****

****

****

****

****

****

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),

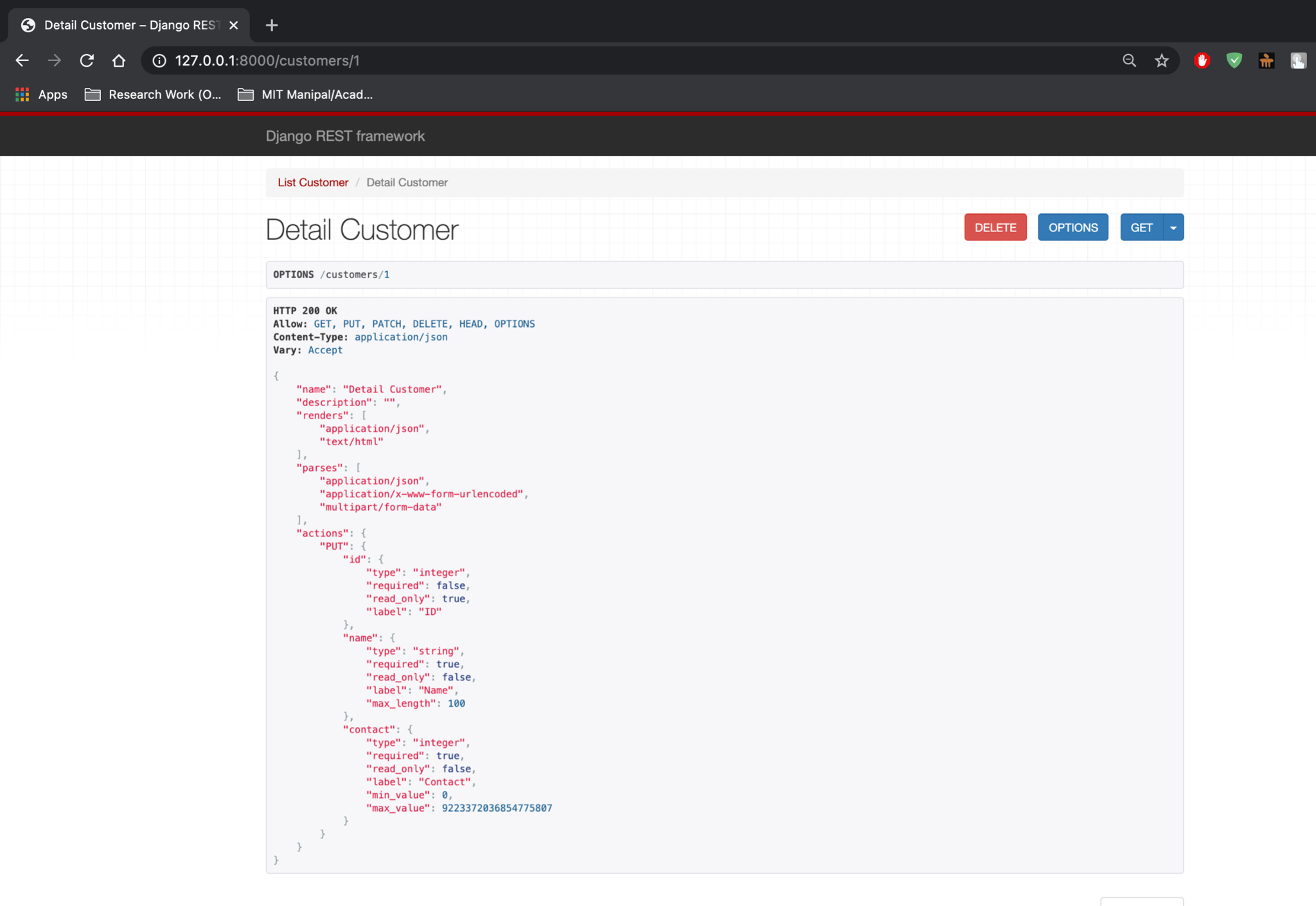
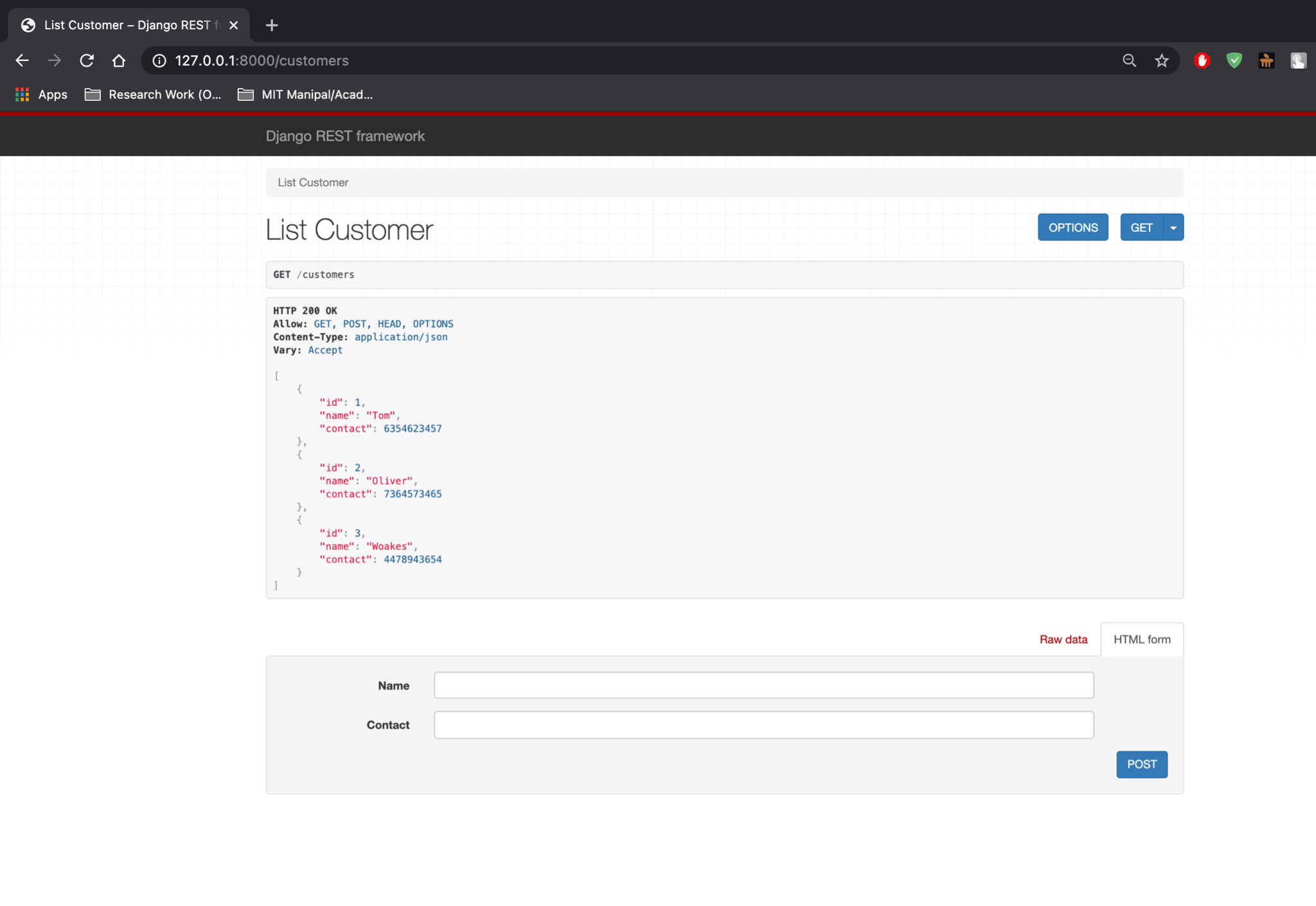
#path('',include("prob1.urls")),

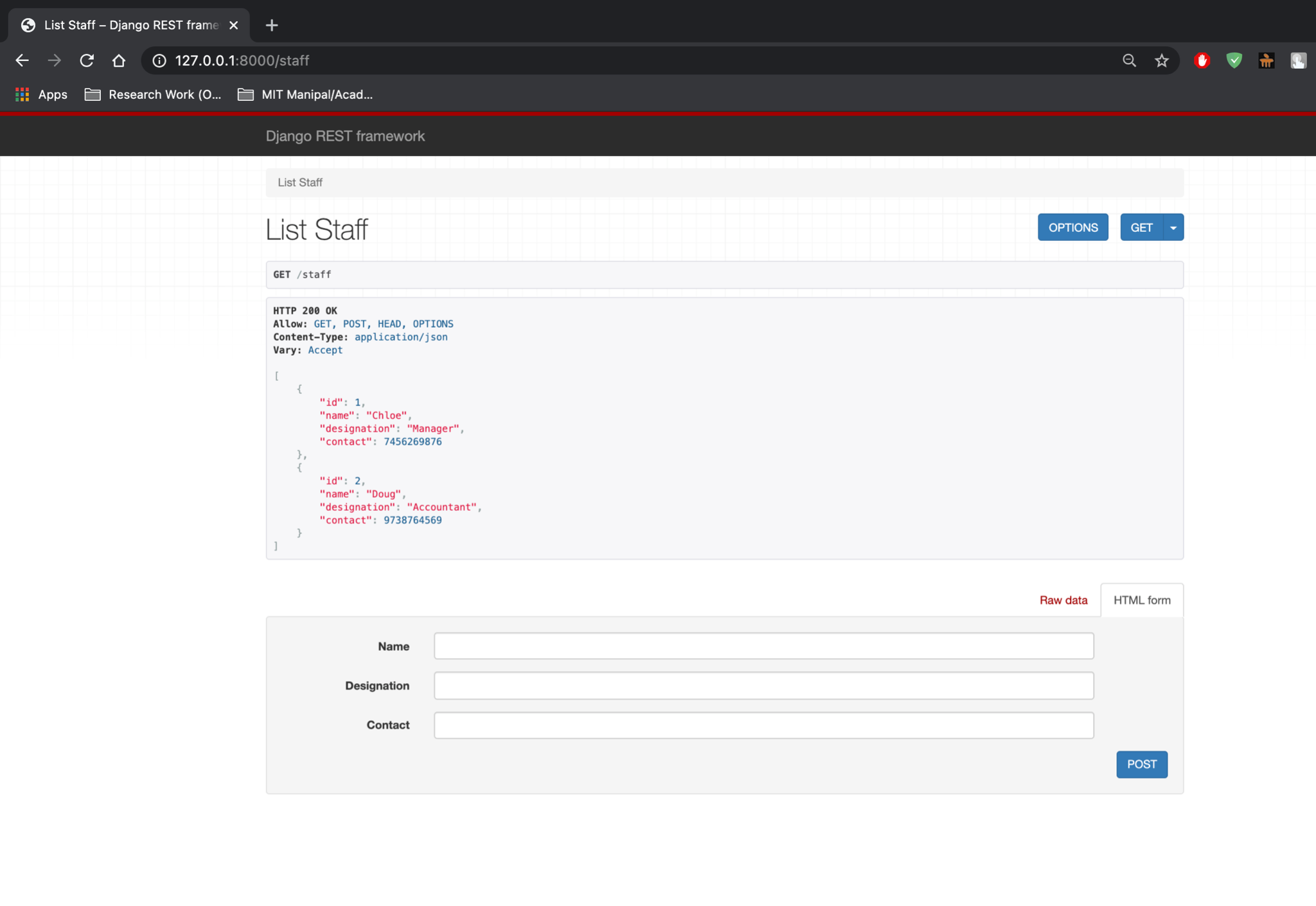
#path('',include("prob2.urls")),

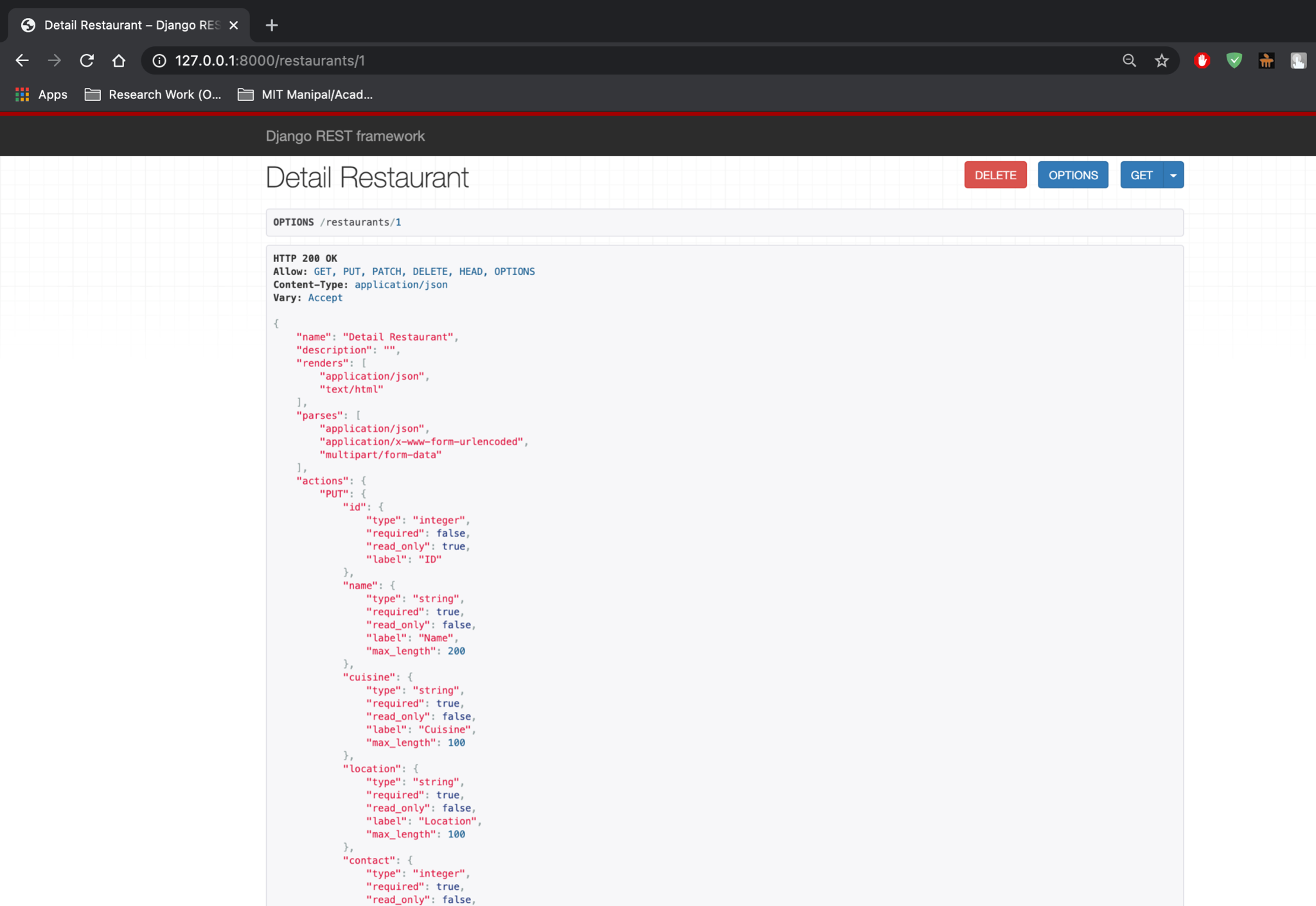
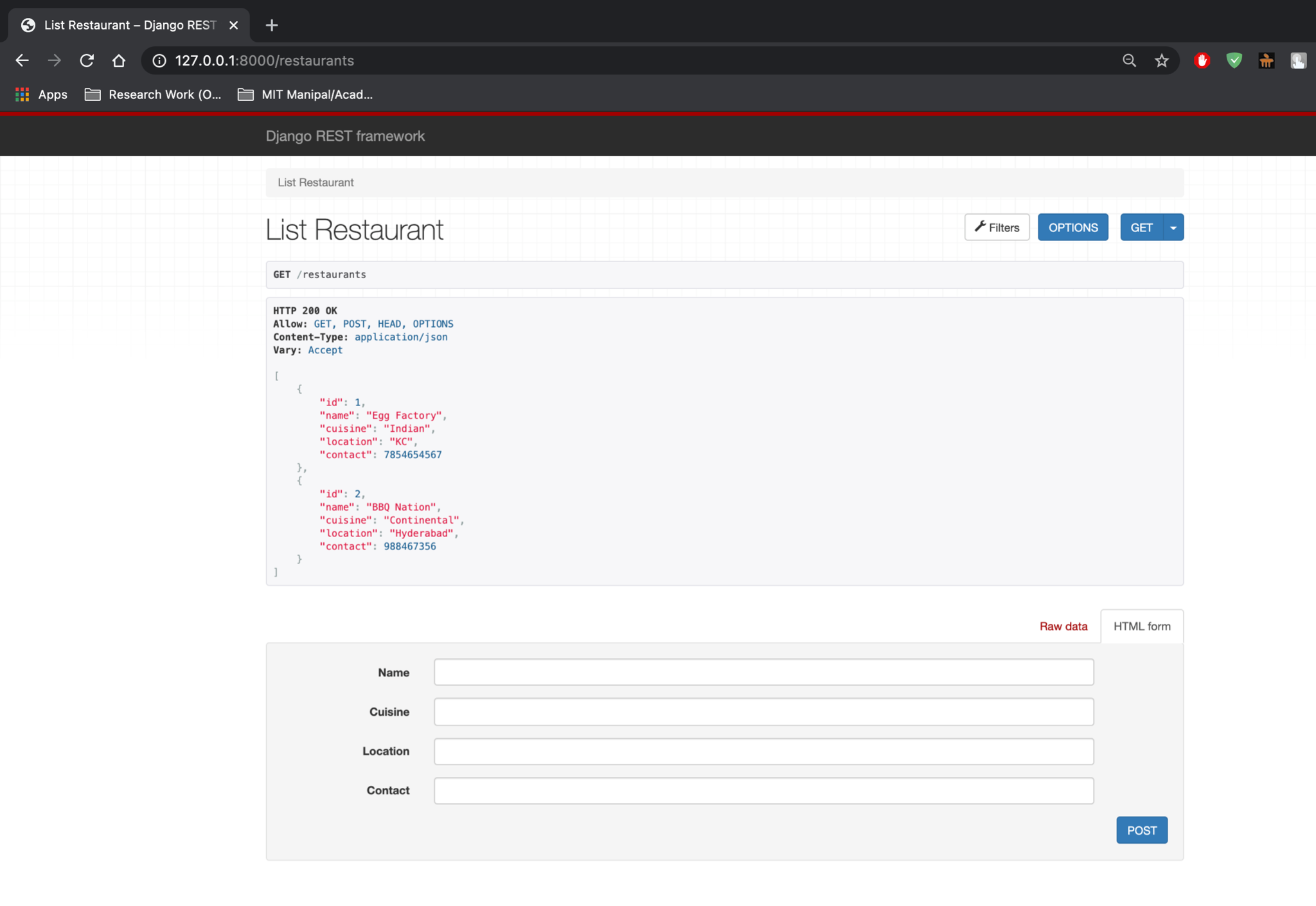
path('',include("prob3.urls")),

#path('',include("prob4.urls")),

]

**Output:**





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:**

