

Cake Delivery System API – Task II Summary

Name: Ishani Bhowmick Group A Team 1

To-dos:

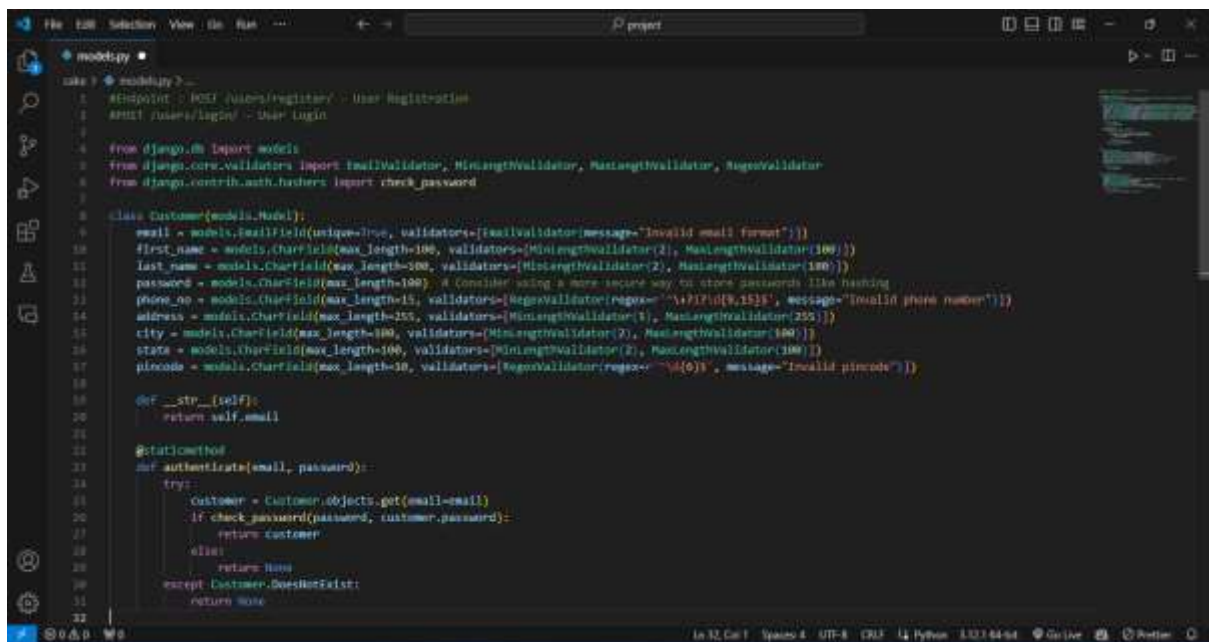
- Make all classes inside a single models.py file instead of separate ones
- Implement the new features given for Task II
- Add error handling as in https codes and validation for inputs

My work:

- Endpoint : POST /users/register/ - User Registration
- POST /users/login/ - User Login
- Cake CRUD APIs
- Cake Customization options CRUD APIs

Programming screenshots

Models.py



```
1 # Endpoint : POST /users/register/ - User Registration
2 # Endpoint : POST /users/login/ - User Login
3
4 from django.db import models
5 from django.core.validators import email_validator, MinLengthValidator, MaxLengthValidator, RegexValidator
6 from django.contrib.auth.hashers import check_password
7
8 class Customer(models.Model):
9     email = models.EmailField(unique=True, validators=[email_validator(message="Invalid email format")])
10     first_name = models.CharField(max_length=100, validators=[MinLengthValidator(2), MaxLengthValidator(100)])
11     last_name = models.CharField(max_length=100, validators=[MinLengthValidator(2), MaxLengthValidator(100)])
12     password = models.CharField(max_length=100) # Consider using a more secure way to store passwords like hashing
13     phone_no = models.CharField(max_length=15, validators=[RegexValidator(regex="^[0-9]{15}$", message="Invalid phone number")])
14     address = models.CharField(max_length=255, validators=[MinLengthValidator(1), MaxLengthValidator(255)])
15     city = models.CharField(max_length=100, validators=[MinLengthValidator(2), MaxLengthValidator(100)])
16     state = models.CharField(max_length=100, validators=[MinLengthValidator(2), MaxLengthValidator(100)])
17     pincode = models.CharField(max_length=10, validators=[RegexValidator(regex="^[0-9]{10}$", message="Invalid pincode")])
18
19     def __str__(self):
20         return self.email
21
22     @staticmethod
23     def authenticate(email, password):
24         try:
25             customer = Customer.objects.get(email=email)
26             if check_password(password, customer.password):
27                 return customer
28             else:
29                 return None
30         except Customer.DoesNotExist:
31             return None
32
```

```

11
12
13 #Cake
14 class Cake(models.Model):
15     name = models.CharField("Name", max_length=100, default='')
16     flavor = models.CharField("Flavor", max_length=100, default='vanilla')
17     size = models.CharField("Size", max_length=50, default='3')
18     price = models.DecimalField("Price", max_digits=10, decimal_places=2, default='500')
19     description = models.TextField("Description", default='')
20     image = models.ImageField(upload_to='cakes/', default='cakes/default.jpg')
21     available = models.BooleanField("Availability", default=True)
22
23     def __str__(self):
24         return self.name
25
26
27 #Cake Customization
28 class CakeCustomization(models.Model):
29     message = models.TextField()
30     egg_version = models.CharField(max_length=10, choices=({'egg', 'eggless'}, {'eggless', 'eggless'}))
31     toppings = models.CharField(max_length=255)
32     shape = models.CharField(max_length=50)
33     cake = models.ForeignKey("Cake", on_delete=models.CASCADE)
34     customer = models.ForeignKey("Customer", on_delete=models.CASCADE)
35
36     def __str__(self):
37         return f'{self.cake.name} Customization'
38

```

Serializers.py

```

1
2 from rest_framework import serializers
3 from .models import Customer, Cake, CakeCustomization
4
5 class CustomerSerializer(serializers.ModelSerializer):
6     class Meta:
7         model = Customer
8         fields = '__all__'
9
10 class CakeSerializer(serializers.ModelSerializer):
11     class Meta:
12         model = Cake
13         fields = '__all__'
14
15 class CakeCustomizationSerializer(serializers.ModelSerializer):
16     class Meta:
17         model = CakeCustomization
18         fields = '__all__'
19

```

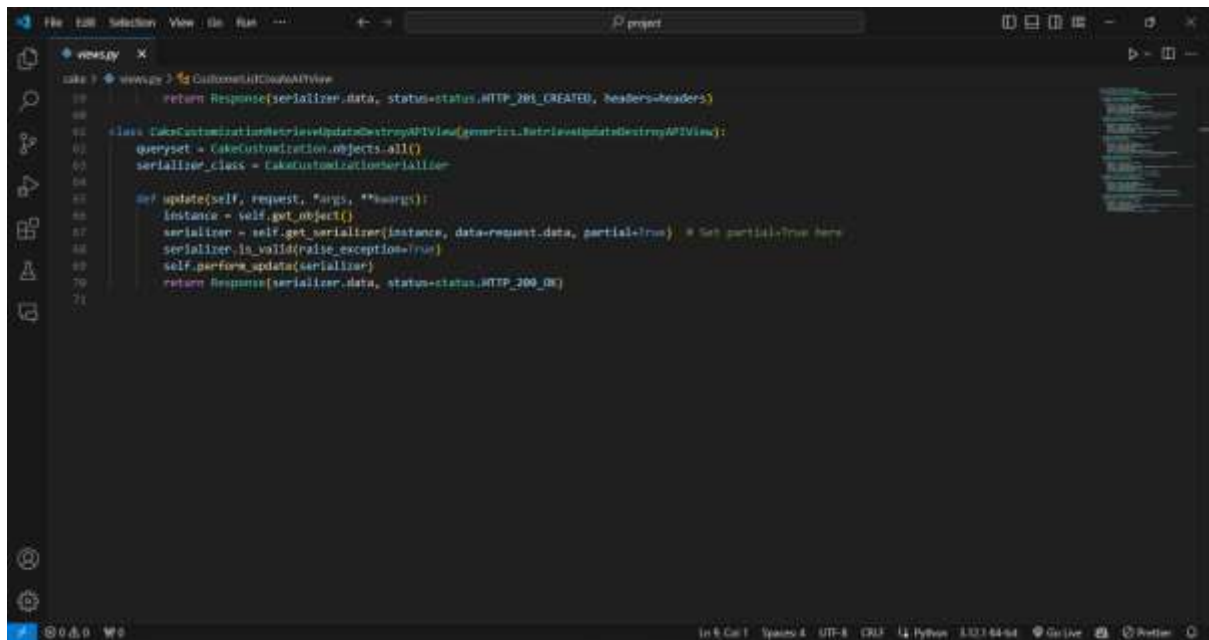
Urls.py

```

1 from django.urls import path
2 from . import views
3
4 urlpatterns = [
5     path('customers/', views.CustomerListCreateAPIView.as_view(), name='customer-list-create'),
6     path('customers/<int:pk>/', views.CustomerRetrieveUpdateDestroyAPIView.as_view(), name='customer-detail'),
7     path('cakes/', views.CakeListCreateAPIView.as_view(), name='cake-list-create'),
8     path('cakes/<int:pk>/', views.CakeRetrieveUpdateDestroyAPIView.as_view(), name='cake-detail'),
9     path('cake-customizations/', views.CakeCustomizationListCreateAPIView.as_view(), name='cake-customization-list-create'),
10    path('cake-customizations/<int:pk>/', views.CakeCustomizationRetrieveUpdateDestroyAPIView.as_view(), name='cake-customization-detail'),
11 ]
12

```

Views.py

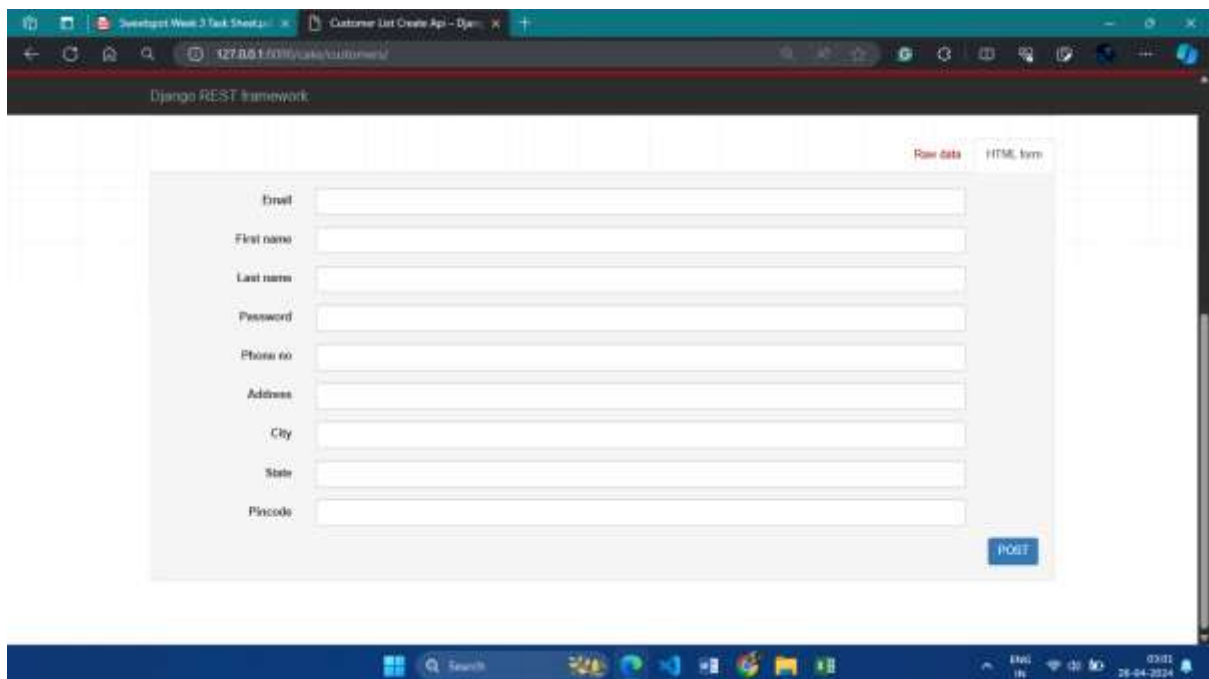


```
19 return Response(serializer.data, status=status.HTTP_201_CREATED, headers=headers)
20
21 class TakeCustomizationRetrieveUpdateDestroyAPIView(generics.RetrieveUpdateDestroyAPIView):
22     queryset = TakeCustomization.objects.all()
23     serializer_class = TakeCustomizationSerializer
24
25     def update(self, request, *args, **kwargs):
26         instance = self.get_object()
27         serializer = self.get_serializer(instance, data=request.data, partial=True)  # Set partial=True here
28         serializer.is_valid(raise_exception=True)
29         self.perform_update(serializer)
30         return Response(serializer.data, status=status.HTTP_200_OK)
```

API testing screenshots

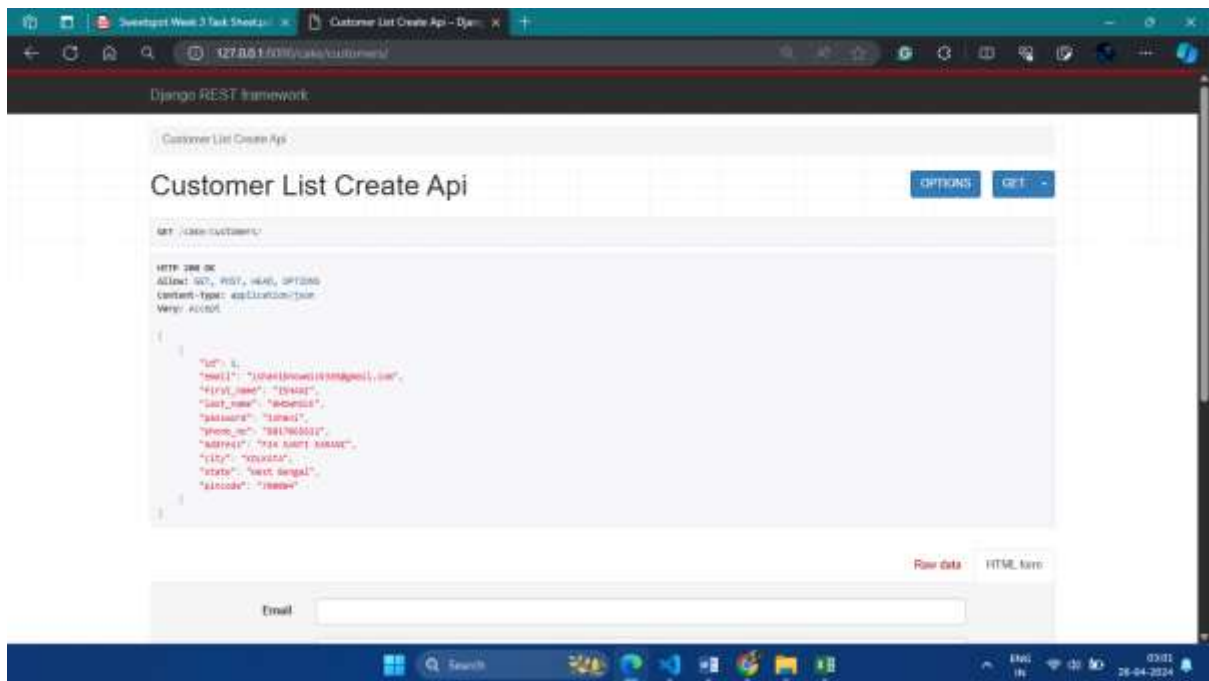
POST /users/register/ - User Registration and Login:

CREATE

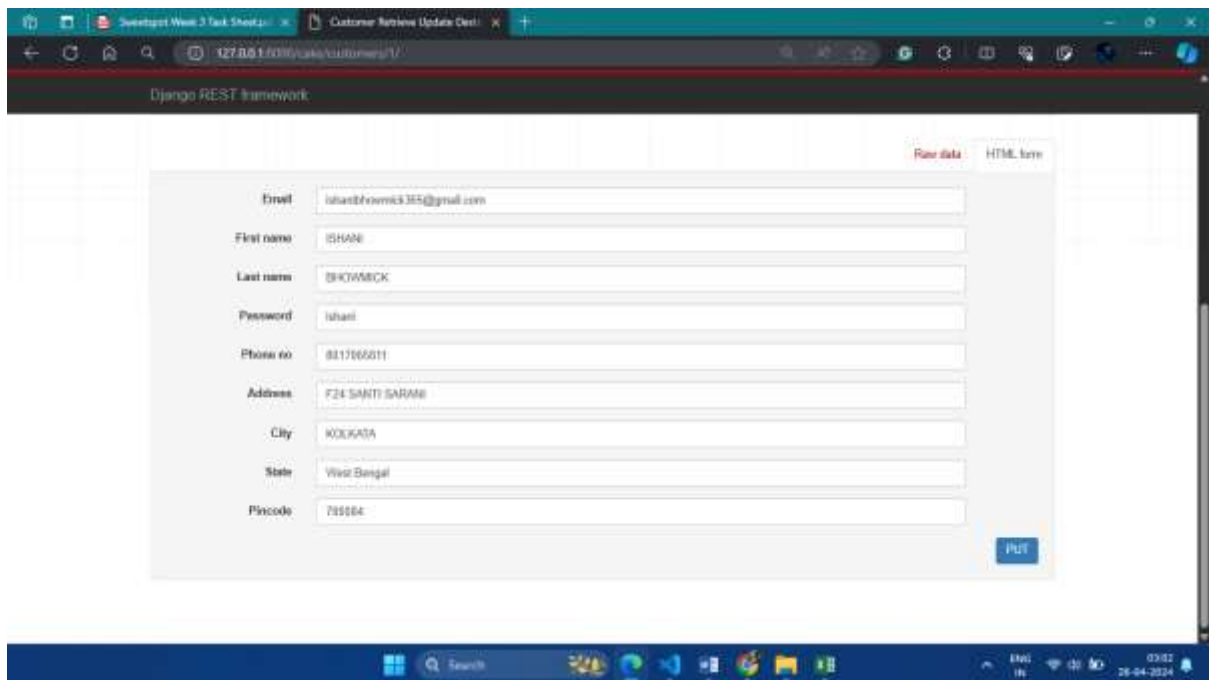


The screenshot shows a web browser window with the URL `127.0.0.1:8000/users/register/`. The page displays a Django REST framework API client interface. At the top, there are tabs for "Raw data" and "HTML form", with "HTML form" currently selected. Below the tabs is a form with the following fields: Email, First name, Last name, Password, Phone no, Address, City, State, and Pincode. Each field has a corresponding input box. At the bottom right of the form, there is a blue button labeled "POST". The browser's address bar shows the URL, and the page title is "Django REST framework".

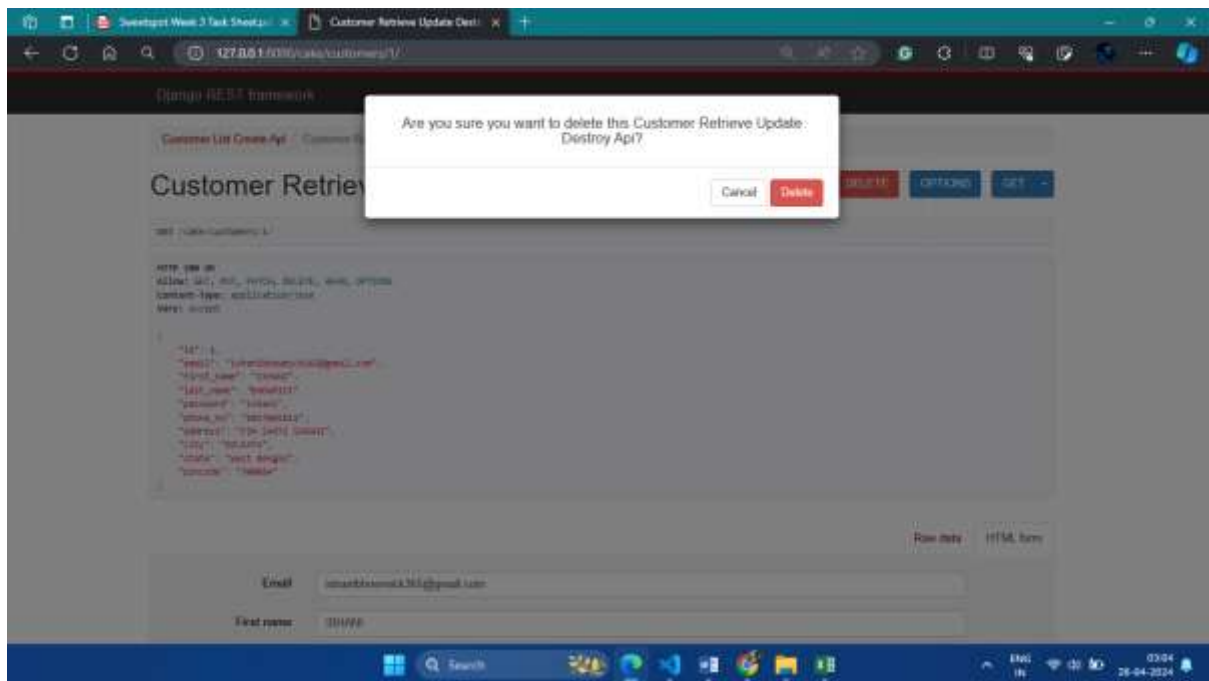
AVAILABLE DATABASE



UPDATE

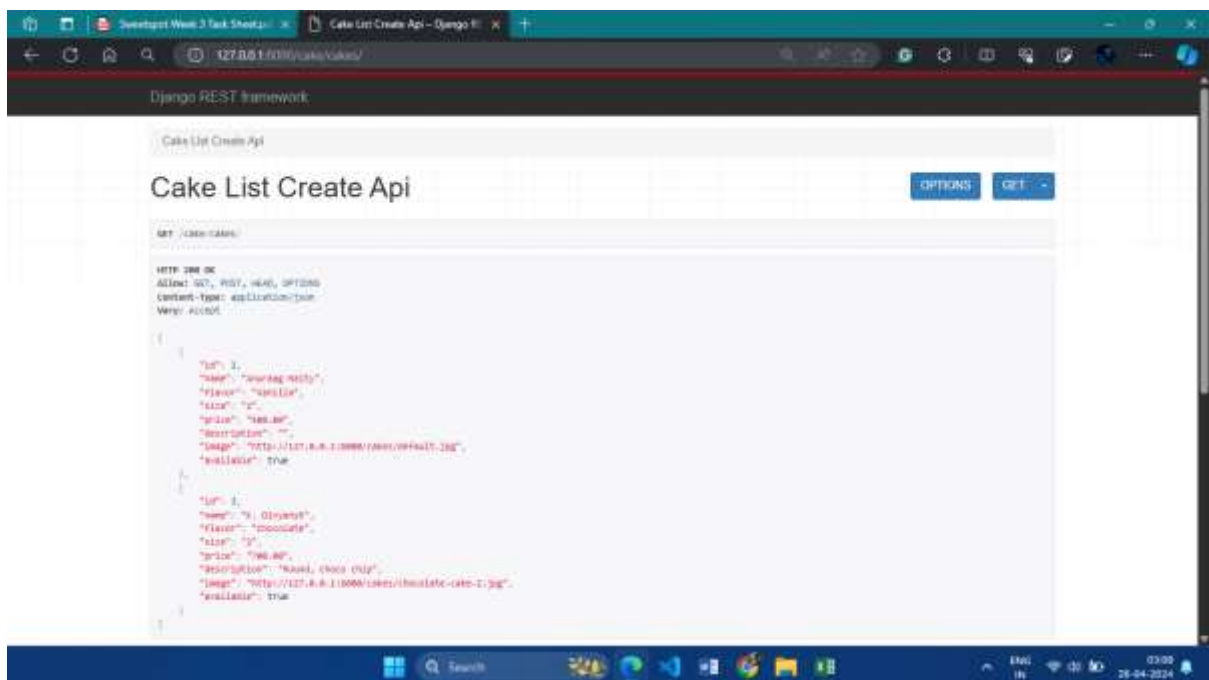


DELETE



Cake CRUD APIs

AVAILABLE DATABASE



CREATE

Django REST framework

```
{  
  "name": "Mango, chocolate",  
  "flavor": "http://127.0.0.1:8080/static/default-cake-2.jpg",  
  "availability": true  
}
```

Raw data HTML form

Name:

Flavor:

Size:

Price:

Description:

Image: (No file chosen)

Availability: ☐

POST

UPDATE

Django REST framework

```
{  
  "name": "Mango",  
  "flavor": "http://127.0.0.1:8080/static/default.jpg",  
  "availability": true  
}
```

Raw data HTML form

Name:

Flavor:

Size:

Price:

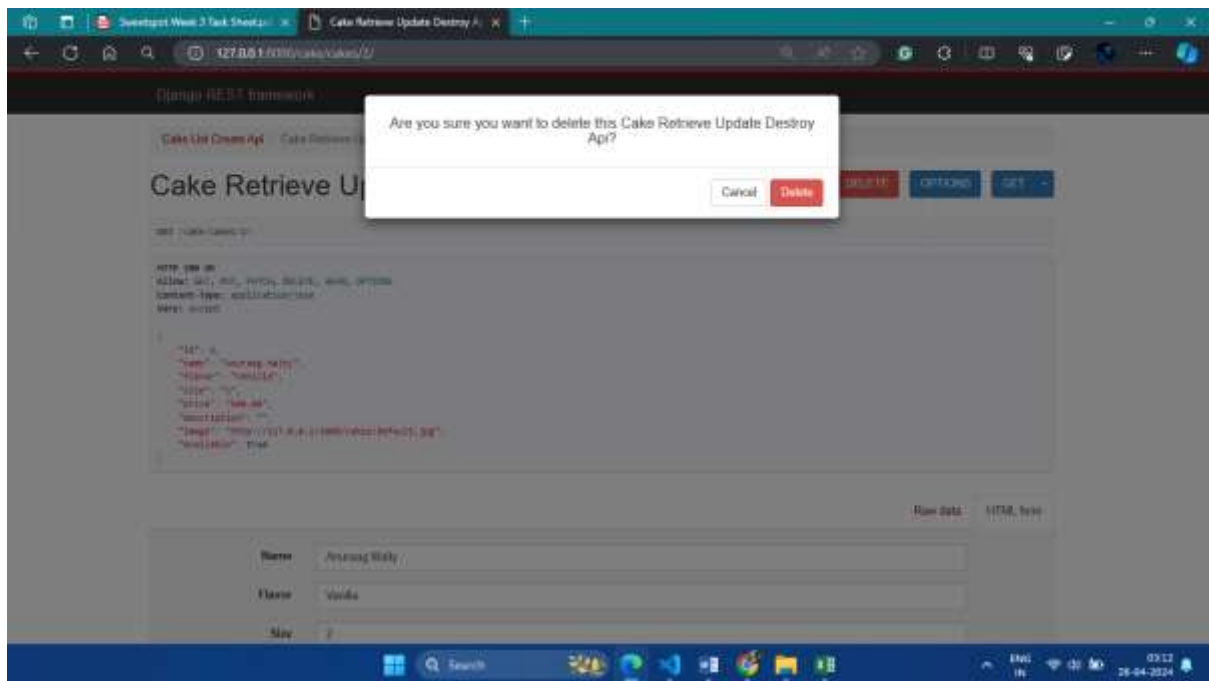
Description:

Image: (No file chosen)

Availability: ☒

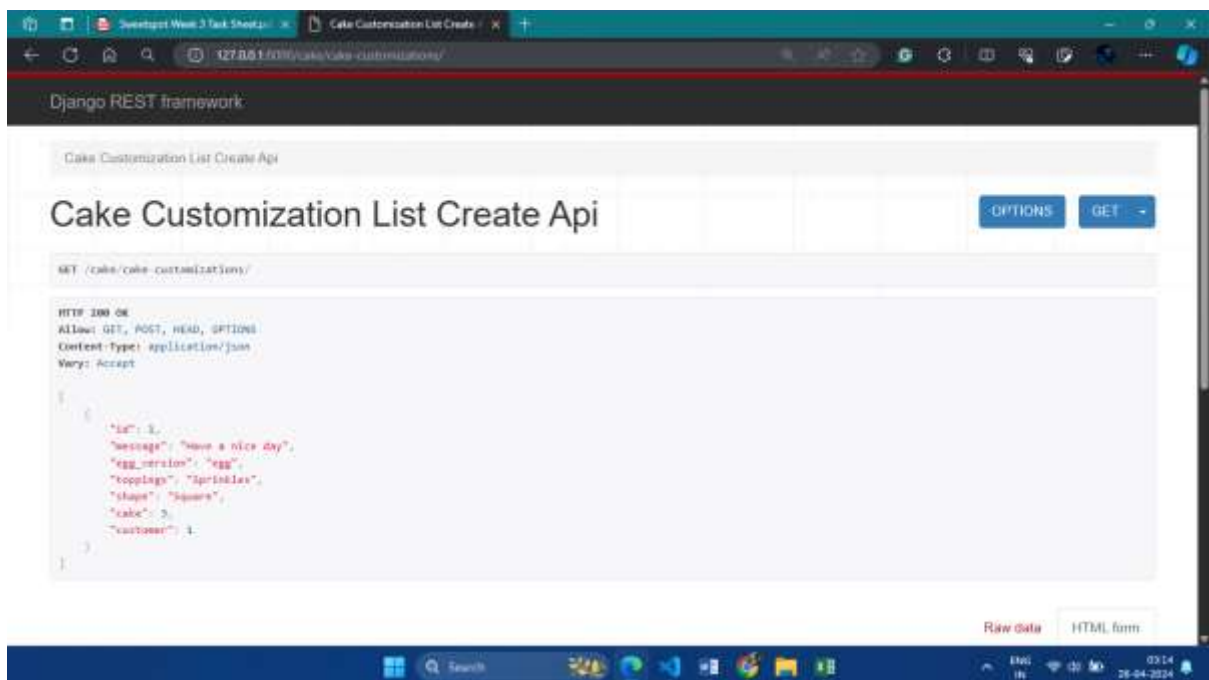
PUT

DELETE



Cake Customization options CRUD APIs

AVAILABLE DATABASE



CREATE

The screenshot shows a web browser window with the URL `127.0.0.1:8080/cake/cake-customizations/`. The page title is "Django REST framework". In the top right corner, there are two tabs: "Raw data" (highlighted in red) and "HTML form". The form contains the following fields:

- Message:** An empty text input field.
- Egg version:** A dropdown menu with "Egg" selected.
- Toppings:** An empty text input field.
- Shape:** An empty text input field.
- Cake:** A dropdown menu with "K. Dnyanesh" selected.
- Customer:** A dropdown menu with "ishanbhowmick365@gmail.com" selected.

A blue "POST" button is located at the bottom right of the form.

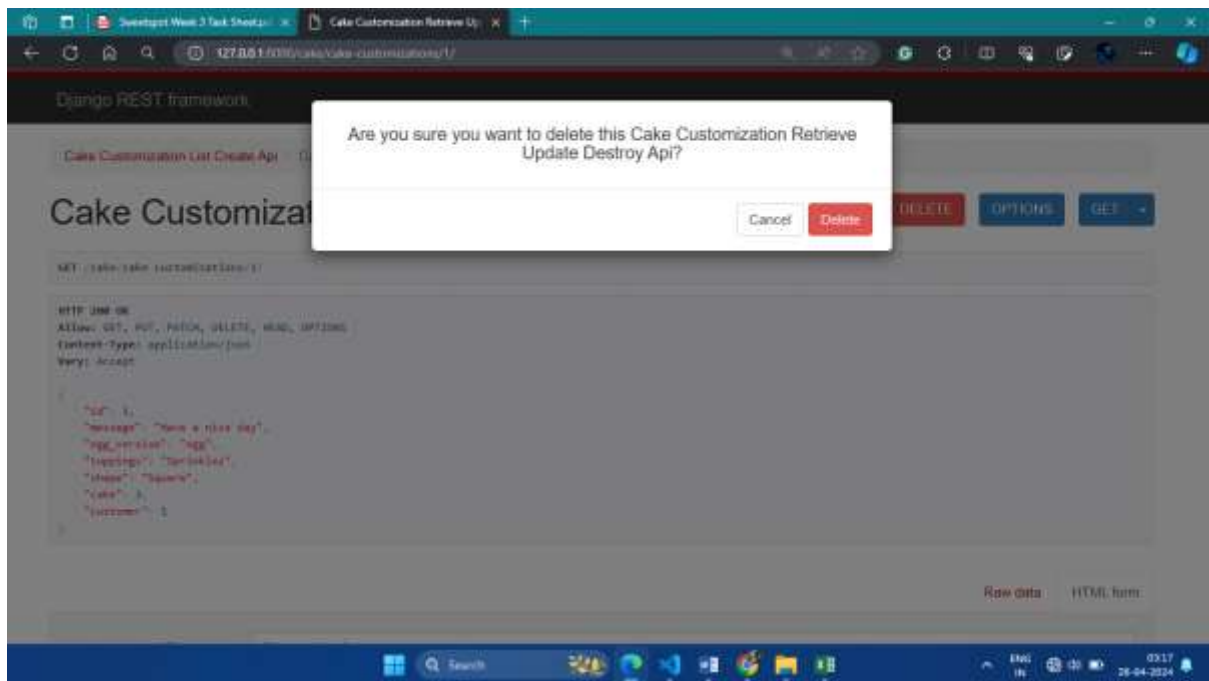
UPDATE

The screenshot shows a web browser window with the URL `127.0.0.1:8080/cake/cake-customizations/1/`. The page title is "Django REST framework". In the top right corner, there are two tabs: "Raw data" (highlighted in red) and "HTML form". The form contains the following fields:

- Message:** A text input field containing "Have a nice day".
- Egg version:** A dropdown menu with "Egg" selected.
- Toppings:** A text input field containing "Sprinkles".
- Shape:** A text input field containing "Square".
- Cake:** A dropdown menu with "K. Dnyanesh" selected.
- Customer:** A dropdown menu with "ishanbhowmick365@gmail.com" selected.

A blue "PUT" button is located at the bottom right of the form.

DELETE



-- end of task --