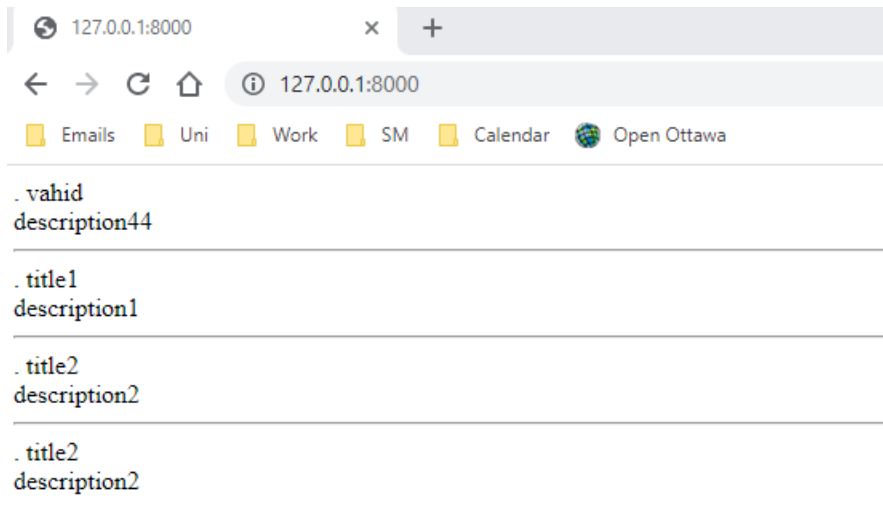
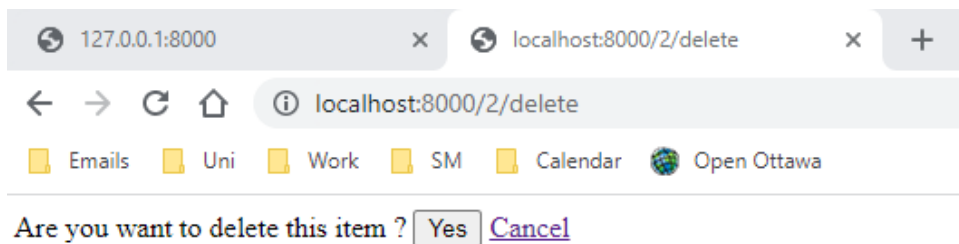


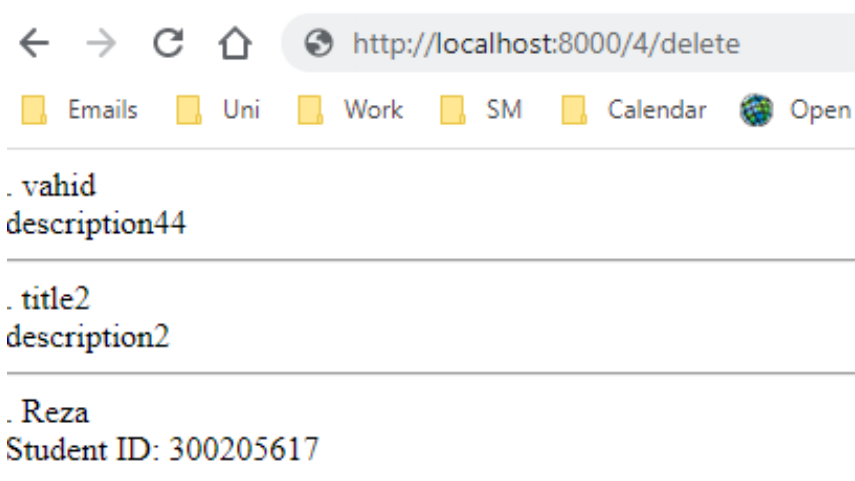
Created some sample test to view the List of items:



Deleting item 2:



The list after deleting:



Updating the vahid title:

← → ↻ 🏠 ⓘ localhost:8000/1/update

📧 Emails 📧 Uni 📧 Work 📧 SM 📧 Calendar 🌐 Open Ottawa

Title:

Description:

description44

Adding new title to the list:

← → ↻ 🏠 ⓘ 127.0.0.1:8000

📧 Emails 📧 Uni 📧 Work 📧 SM 📧 Calendar 🌐 Open Ot

Title:

Description:

Role: TA

The last update and added new titles to the existing list:

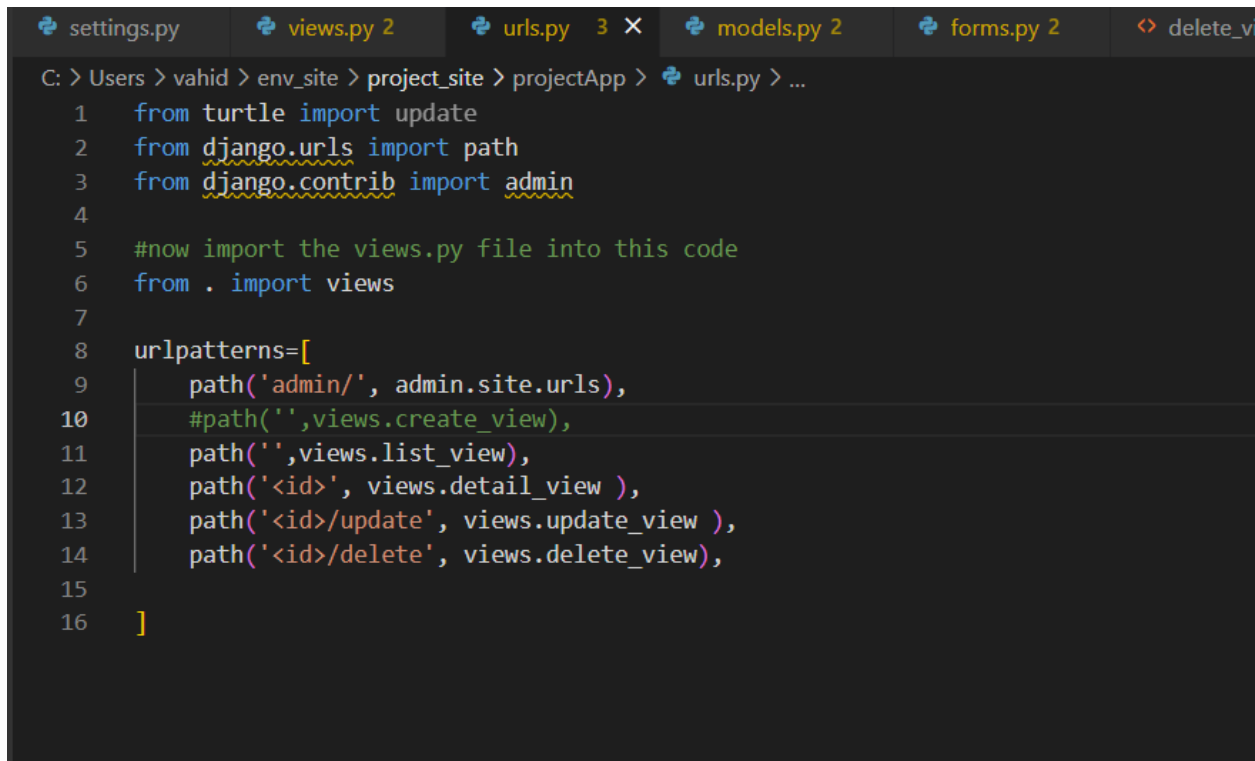
🌐 127.0.0.1:8000 × 🌐 localhost:8000 × +

← → ↻ 🏠 ⓘ localhost:8000

📧 Emails 📧 Uni 📧 Work 📧 SM 📧 Calendar 🌐 Open Ottawa

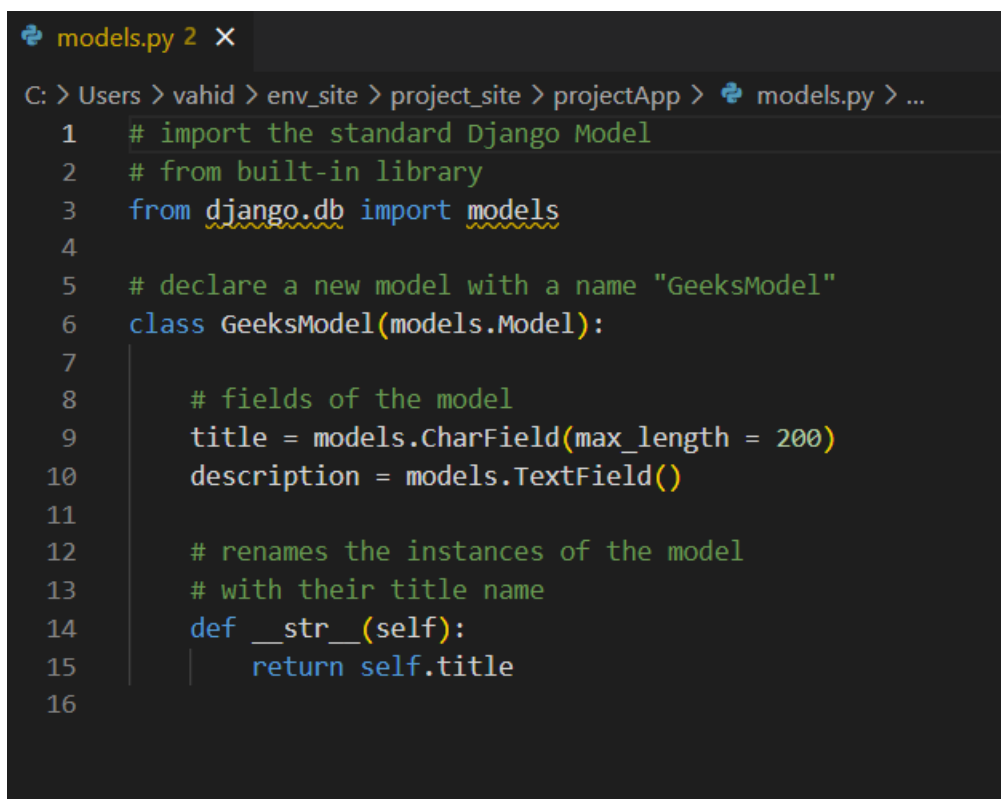
. vahid
Student ID: 300204712
. Reza
Student ID: 300205617
. Rohit
Role: TA
. Masoud
Role: Professor

To see the list we should avoid creat functionality in the urls.py as follows:



```
settings.py  views.py 2  urls.py 3 X  models.py 2  forms.py 2  delete_v...
C: > Users > vahid > env_site > project_site > projectApp > urls.py > ...
1  from turtle import update
2  from django.urls import path
3  from django.contrib import admin
4
5  #now import the views.py file into this code
6  from . import views
7
8  urlpatterns=[
9      path('admin/', admin.site.urls),
10     #path('',views.create_view),
11     path('',views.list_view),
12     path('<id>', views.detail_view ),
13     path('<id>/update', views.update_view ),
14     path('<id>/delete', views.delete_view),
15
16 ]
```

The coding behind can be viewed as follows:



```
models.py 2 X
C: > Users > vahid > env_site > project_site > projectApp > models.py > ...
1  # import the standard Django Model
2  # from built-in library
3  from django.db import models
4
5  # declare a new model with a name "GeeksModel"
6  class GeeksModel(models.Model):
7
8      # fields of the model
9      title = models.CharField(max_length = 200)
10     description = models.TextField()
11
12     # renames the instances of the model
13     # with their title name
14     def __str__(self):
15         return self.title
16
```

```
forms.py 2 X
C: > Users > vahid > env_site > project_site > projectApp > forms.py > ...
1  from django import forms
2  from .models import GeeksModel
3
4
5  # creating a form
6  class GeeksForm(forms.ModelForm):
7
8      # create meta class
9      class Meta:
10         # specify model to be used
11         model = GeeksModel
12
13         # specify fields to be used
14         fields = [
15             "title",
16             "description",
17         ]
18
```

```
create_view.html X
C: > Users > vahid > env_site > project_site > projectApp > templates > create_view.html > form
1  <form method="POST" enctype="multipart/form-data">
2
3      <!-- Security token -->
4      {% csrf_token %}
5
6      <!-- Using the formset -->
7      {{ form.as_p }}
8
9      <input type="submit" value="Submit">
10 </form>
```

```
<> delete_view.html X
C: > Users > vahid > env_site > project_site > projectApp > templates > <> delete_view.html > div.main
1  <div class="main">
2      <!-- Create a Form -->
3      <form method="POST">
4          <!-- Security token by Django -->
5          {% csrf_token %}
6          Are you want to delete this item ?
7          <input type="submit" value="Yes" />
8          <a href="/">Cancel </a>
9      </form>
10 </div>
```

```
<> list_view.html X
C: > Users > vahid > env_site > project_site > projectApp > templates > <> list_view.html > div.main
1  <div class="main">
2
3      {% for data in dataset %}.
4
5      {{ data.title }}<br/>
6      {{ data.description }}<br/>
7      <hr/>
8
9      {% endfor %}
10
11 </div>
```

```
<> update_view.html X
C: > Users > vahid > env_site > project_site > projectApp > templates > <> update_view.html > div.main
1  <div class="main">
2      <!-- Create a Form -->
3      <form method="POST">
4          <!-- Security token by Django -->
5          {% csrf_token %}
6
7          <!-- form as paragraph -->
8          {{ form.as_p }}
9
10         <input type="submit" value="Update">
11     </form>
12
13 </div>
```

```
views.py 2 X
C: > Users > vahid > env_site > project_site > projectApp > views.py > ...
1  from django.shortcuts import render
2
3  # relative import of forms
4  from .models import GeeksModel
5  from .forms import GeeksForm
6  from django.shortcuts import (get_object_or_404,
7                                render,
8                                HttpResponseRedirect)
9
10
11
12 def create_view(request):
13     # dictionary for initial data with
14     # field names as keys
15     context = {}
16
17     # add the dictionary during initialization
18     form = GeeksForm(request.POST or None)
19     if form.is_valid():
20         form.save()
21
22     context['form'] = form
23     return render(request, "create_view.html", context)
24
25
26
27 def list_view(request):
28     # dictionary for initial data with
29     # field names as keys
30     context = {}
31
32     # add the dictionary during initialization
33     context["dataset"] = GeeksModel.objects.all()
34
35     return render(request, "list_view.html", context)
```

```
37 def detail_view(request, id):
38     # dictionary for initial data with
39     # field names as keys
40     context = {}
41
42     # add the dictionary during initialization
43     context["data"] = GeeksModel.objects.get(id = id)
44
45     return render(request, "detail_view.html", context)
46
47 # update view for details
48 def update_view(request, id):
49     # dictionary for initial data with
50     # field names as keys
51     context = {}
52
53     # fetch the object related to passed id
54     obj = get_object_or_404(GeeksModel, id = id)
55
56     # pass the object as instance in form
57     form = GeeksForm(request.POST or None, instance = obj)
58
59     # save the data from the form and
60     # redirect to detail_view
61     if form.is_valid():
62         form.save()
63         return HttpResponseRedirect("/") + id
64
65     # add form dictionary to context
66     context["form"] = form
67
68     return render(request, "update_view.html", context)
```

```
69
70 # from django.shortcuts import (get_object_or_404,
71 #                                render,
72 #                                HttpResponseRedirect)
73
74 # from .models import GeeksModel
75
76
77 # delete view for details
78 def delete_view(request, id):
79     # dictionary for initial data with
80     # field names as keys
81     context = {}
82
83     # fetch the object related to passed id
84     obj = get_object_or_404(GeeksModel, id = id)
85
86
87     if request.method == "POST":
88         # delete object
89         obj.delete()
90         # after deleting redirect to
91         # home page
92         return HttpResponseRedirect("/")
93
94     return render(request, "delete_view.html", context)
95
```