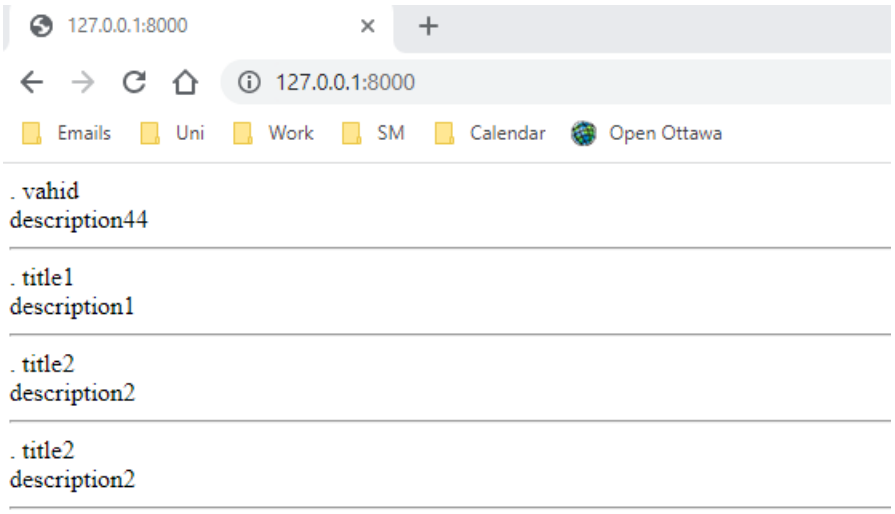
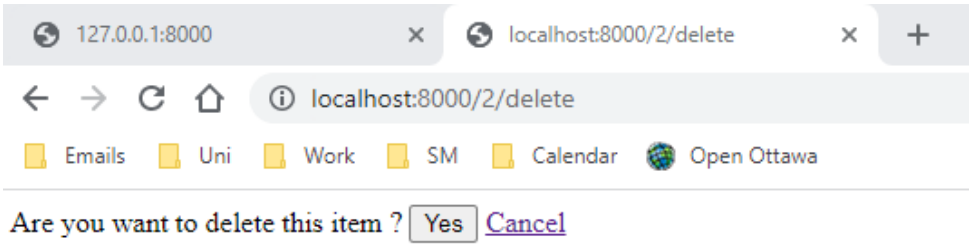
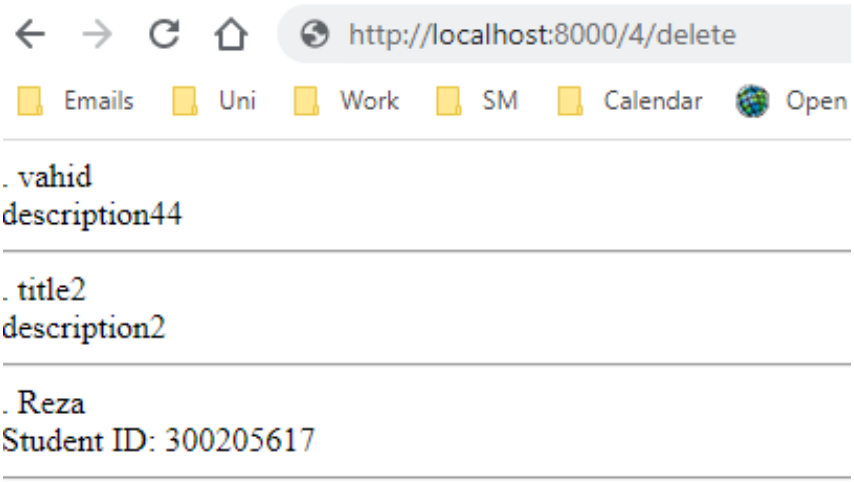
Created some sample test to view the List of items:



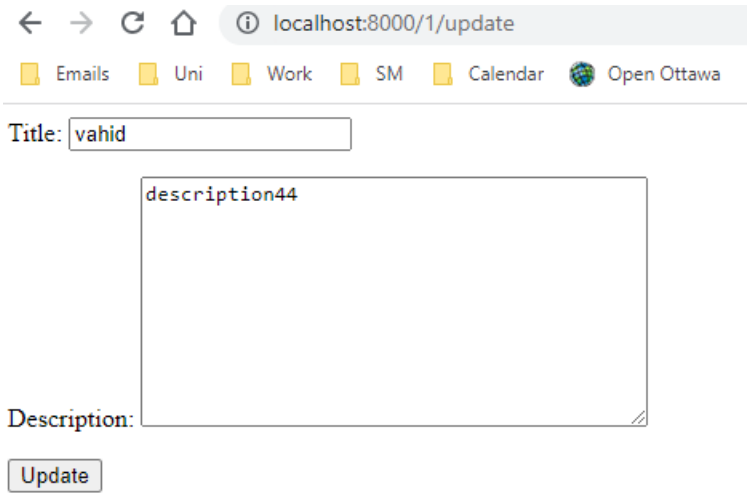
Deleting item 2:



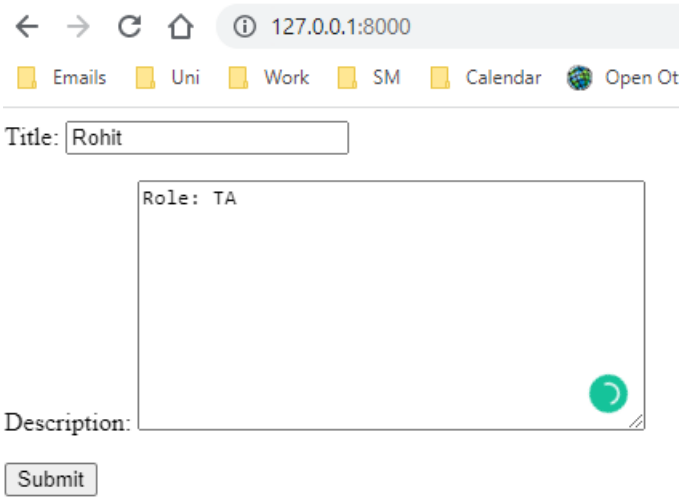
The list after deleting:



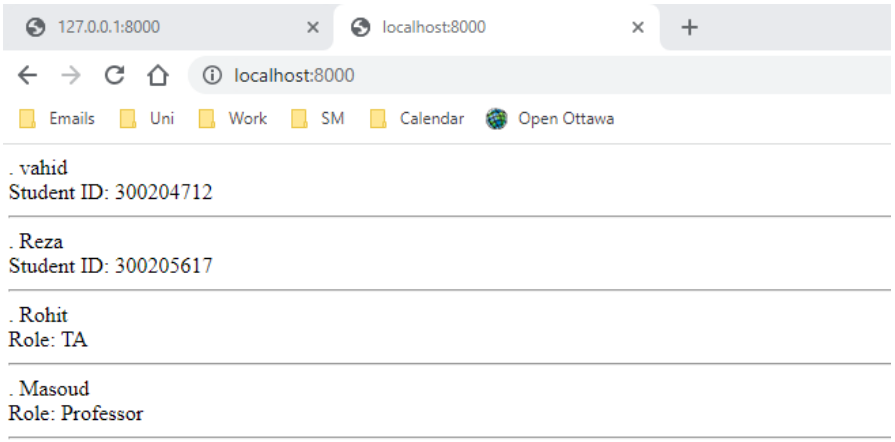
Updating the vahid title:



Adding new title to the list:



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



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

```
settings.py   views.py 2   urls.py 3 ×   models.py 2   forms.py 2   delete_v
C: > Users > vahid > env_site > project_site > projectApp > urls.py > ...
from turtle import update
from django.urls import path
from django.contrib import admin

#now import the views.py file into this code
from . import views

urlpatterns=[
    path('admin/', admin.site.urls),
    #path('',views.create_view),
    path('',views.list_view),
    path('<id>', views.detail_view ),
    path('<id>/update', views.update_view ),
    path('<id>/delete', views.delete_view),

]
```

The coding behind can be viewed as follows:

```
models.py 2 ×
C: > Users > vahid > env_site > project_site > projectApp > models.py > ...
# import the standard Django Model
# from built-in library
from django.db import models

# declare a new model with a name "GeeksModel"
class GeeksModel(models.Model):

    # fields of the model
    title = models.CharField(max_length = 200)
    description = models.TextField()

    # renames the instances of the model
    # with their title name
    def __str__(self):
        return self.title

```

forms.py 2

C: > Users > vahid > env_site > project_site > projectApp > forms.py > ...

```python
from django import forms
from .models import GeeksModel


# creating a form
class GeeksForm(forms.ModelForm):

    # create meta class
    class Meta:
        # specify model to be used
        model = GeeksModel

        # specify fields to be used
        fields = [
            "title",
            "description",
        ]

```

create_view.html

C: > Users > vahid > env_site > project_site > projectApp > templates > create_view.html > form

```html
<form method="POST" enctype="multipart/form-data">

    <!-- Security token -->
    {% csrf_token %}

    <!-- Using the formset -->
    {{ form.as_p }}

    <input type="submit" value="Submit">
</form>
```

delete_view.html

C: > Users > vahid > env_site > project_site > projectApp > templates > delete_view.html > div.main

```
<div class="main">
    <!-- Create a Form -->
    <form method="POST">
        <!-- Security token by Django -->
        {% csrf_token %}
        Are you want to delete this item ?
        <input type="submit" value="Yes" />
        <a href="/">Cancel </a>
    </form>
</div>
```

list_view.html

C: > Users > vahid > env_site > project_site > projectApp > templates > list_view.html > div.main

```
<div class="main">

    {% for data in dataset %}.

    {{ data.title }}<br/>
    {{ data.description }}<br/>
    <hr/>

    {% endfor %}

</div>
```

update_view.html

: > Users > vahid > env_site > project_site > projectApp > templates > update_view.html > div.main

```
<div class="main">
    <!-- Create a Form -->
    <form method="POST">
        <!-- Security token by Django -->
        {% csrf_token %}

        <!-- form as paragraph -->
        {{ form.as_p }}

        <input type="submit" value="Update">
    </form>

</div>
```

views.py 2 ×

C: > Users > vahid > env_site > project_site > projectApp > views.py > ...

```python
from django.shortcuts import render

# relative import of forms
from .models import GeeksModel
from .forms import GeeksForm
from django.shortcuts import (get_object_or_404,
                              render,
                              HttpResponseRedirect)



def create_view(request):
    # dictionary for initial data with
    # field names as keys
    context ={}

    # add the dictionary during initialization
    form = GeeksForm(request.POST or None)
    if form.is_valid():
        form.save()

    context['form']= form
    return render(request, "create_view.html", context)



def list_view(request):
    # dictionary for initial data with
    # field names as keys
    context ={}

    # add the dictionary during initialization
    context["dataset"] = GeeksModel.objects.all()

    return render(request, "list_view.html", context)
```

```python
def detail_view(request, id):
    # dictionary for initial data with
    # field names as keys
    context ={}

    # add the dictionary during initialization
    context["data"] = GeeksModel.objects.get(id = id)

    return render(request, "detail_view.html", context)

# update view for details
def update_view(request, id):
    # dictionary for initial data with
    # field names as keys
    context ={}

    # fetch the object related to passed id
    obj = get_object_or_404(GeeksModel, id = id)

    # pass the object as instance in form
    form = GeeksForm(request.POST or None, instance = obj)

    # save the data from the form and
    # redirect to detail_view
    if form.is_valid():
        form.save()
        return HttpResponseRedirect("/"+id)

    # add form dictionary to context
    context["form"] = form

    return render(request, "update_view.html", context)
```

```python

# from django.shortcuts import (get_object_or_404,
#                               render,
#                               HttpResponseRedirect)

# from .models import GeeksModel


# delete view for details
def delete_view(request, id):
    # dictionary for initial data with
    # field names as keys
    context ={}

    # fetch the object related to passed id
    obj = get_object_or_404(GeeksModel, id = id)


    if request.method =="POST":
        # delete object
        obj.delete()
        # after deleting redirect to
        # home page
        return HttpResponseRedirect("/")

    return render(request, "delete_view.html", context)

```