MODELS.PY

from django.db import models

from django.db.models.signals import pre\_save

from django.dispatch import receiver

from student.validations import \*

from student.choices import \*

from student.defaults import \*

# Create your models here.

class Groups(models.Model):

#Group

Group = models.CharField(default = '',max\_length = 10, primary\_key = True)

Class = models.CharField(default = '',max\_length = 10)

Groups\_Type = models.CharField(default = '',max\_length=10, choices = Groups\_Type\_Choices)

class Student(models.Model):

#TIMESTAMP

Date\_Time = models.DateTimeField(auto\_now = True)

Year\_Of\_Admission = models.IntegerField(default = Year\_Admission)

#STUDENT DETAILS

First\_Name = models.CharField(max\_length = 50)

Middle\_Name = models.CharField(max\_length = 50, blank = True)

Last\_Name = models.CharField(max\_length = 50)

Class = models.CharField(max\_length = 50)

Group = models.ForeignKey(Groups,on\_delete = models.PROTECT)

Admission\_Number = models.CharField(max\_length = 50)

Roll\_Number = models.CharField(max\_length = 50, primary\_key = True)

Date\_of\_Birth = models.CharField(max\_length = 10,help\_text = 'YYYY-MM-DD')

Gender = models.CharField(max\_length= 20, choices = Gender\_Choices)

Father\_Name = models.CharField(max\_length = 50, blank = True, null = True)

Mother\_Name = models.CharField(max\_length = 50, blank = True, null = True)

Guardian\_Name = models.CharField(max\_length = 50, blank = True, null = True, default = '')

Religion = models.CharField(max\_length = 50)

Nationality = models.CharField(max\_length = 50)

Aadhaar\_Number = models.CharField(max\_length = 50, default = '')

#CONTACT DETAILS

Mobile\_Number = models.CharField(max\_length = 10, validators = [Phone\_Number])

Email\_Id = models.EmailField(max\_length = 250)

Alternate\_Mobile\_Number = models.CharField(max\_length = 10, validators = [Phone\_Number], null = True)

#ADDRESS DETAILS

Door\_Number = models.CharField(max\_length = 20, help\_text = 'Flat/Door/Block Number')

Premises = models.CharField(max\_length = 100, help\_text = 'Name of the Premises/Building/Village')

Street = models.CharField(max\_length = 100, help\_text = 'Street/Road/Post Office', blank = True)

Area = models.CharField(max\_length = 50, help\_text = 'Area/Locality Details',default = "", blank = True)

City = models.CharField(max\_length = 50, default = "", blank = True)

State = models.CharField(max\_length = 50, default = "")

Country = models.CharField(max\_length = 50, default = '')

Pin\_Code = models.CharField(max\_length = 20, default = "")

Student\_Image = models.ImageField(default = "",upload\_to = 'Images/')

Study\_Of\_Conduct = models.ImageField(default = "",upload\_to = 'Images/',validators = [Image\_Size],help\_text='Size should be less than 300 KB')

Transfer\_Certificate = models.ImageField(default = "",upload\_to = 'Images/',validators = [Image\_Size],help\_text='Size should be less than 300 KB')

Tenth\_Marks\_Long\_Memo = models.ImageField(default = "",upload\_to = 'Images/',validators = [Image\_Size],help\_text='Size should be less than 300 KB')

Intermediate\_Marks\_Short\_Memo = models.ImageField(default = "",upload\_to = 'Images/',validators = [Image\_Size],help\_text='Size should be less than 300 KB')

Student\_Signature = models.ImageField(default = "",upload\_to = 'Images/',validators = [Image\_Size],help\_text='Size should be less than 300 KB')

valid\_User\_Password = models.CharField(max\_length = 20,editable = False)

@receiver(pre\_save, sender=Student)

def default\_Student(sender, instance, \*\*kwargs):

if not instance.valid\_User\_Password:

instance.valid\_User\_Password = instance.Roll\_Number

class Courses(models.Model):

#Course Details

Course\_Code = models.CharField(max\_length = 10, primary\_key = True)

Course\_Subject = models.CharField(max\_length = 20, default = '')

Course\_Title = models.CharField(max\_length = 100,default = '')

class Department(models.Model):

#Department Details

Department\_Name = models.CharField(max\_length = 30)

Department\_Id = models.CharField(max\_length = 10, primary\_key = True)

class Teacher(models.Model):

#Teacher Details

Teacher\_Name = models.CharField(max\_length = 50)

Teacher\_Id = models.CharField(max\_length = 10, primary\_key = True)

Teacher\_Mobile\_Number = models.CharField(max\_length = 10, validators = [Phone\_Number])

Tecaher\_Email\_Id = models.EmailField(max\_length = 250)

Teacher\_Department\_Id = models.ForeignKey(Department, on\_delete = models.CASCADE)

valid\_User\_Password\_Teacher = models.CharField(default = '',max\_length = 20,editable = False)

@receiver(pre\_save, sender=Teacher)

def default\_Teacher(sender, instance, \*\*kwargs):

if not instance.valid\_User\_Password\_Teacher:

instance.valid\_User\_Password\_Teacher = instance.Teacher\_Id

class Student\_Marks(models.Model):

#Marks Alloted

Date\_Time\_Marks = models.DateTimeField(auto\_now = True)

Semester = models.CharField(max\_length= 20, choices = Semester\_Choices, null = True)

Marks\_Type = models.CharField(max\_length = 30, choices = Exam\_Type\_Choices)

Course\_Code = models.ForeignKey(Courses,on\_delete = models.PROTECT, default = '')

Roll\_Number\_Marks = models.ForeignKey(Student,on\_delete = models.CASCADE, default = '')

Teacher\_Id\_Marks = models.ForeignKey(Teacher,on\_delete = models.PROTECT, default = '')

Marks\_Total = models.CharField(max\_length = 3,default = '')

Marks\_Alloted = models.DecimalField(max\_digits = 6, decimal\_places = 2,null = True)

class Student\_Attendance(models.Model):

#Attendance Given

Date\_Time\_Marks = models.DateTimeField(auto\_now = True)

Date\_Time\_Original = models.DateTimeField(default = '')

Semester = models.CharField(max\_length= 20, choices = Semester\_Choices, null = True)

Course\_Code = models.ForeignKey(Courses,on\_delete = models.PROTECT, default = '')

Roll\_Number\_Attendance = models.ForeignKey(Student,on\_delete = models.CASCADE ,default = '')

Teacher\_Id\_Attendance = models.ForeignKey(Teacher,on\_delete = models.PROTECT, default = '')

Attendance\_Date = models.CharField(max\_length = 15, choices = Attendance\_Choices)

class Teacher\_Course(models.Model):

#Courses and Teacher

Group = models.ForeignKey(Groups, on\_delete = models.PROTECT)

Teacher\_Id = models.ForeignKey(Teacher,on\_delete = models.PROTECT)

Course\_Code = models.ForeignKey(Courses,on\_delete = models.PROTECT)

Semester\_Teacher = models.CharField(max\_length= 20, choices = Semester\_Choices)

Year\_Of\_Admission = models.IntegerField(default = Year\_Admission)

MODEL CHOICES.PY

Gender\_Choices = (("Male","Male"),

("Female","Female"),

("Others","Others"))

Semester\_Choices = (("1","1"),

("2","2"),

("3","3"),

("4","4"),

("5","5"),

("6","6"))

Exam\_Type\_Choices = (("I Internal","I Internal"),

("II Internal","II Internal"),

("Semester","Semester"),

("Assignment","Assignment"),

("Attendance","Attendance"))

Attendance\_Choices = (("PRESENT","PRESENT"),

("ABSENT","ABSENT"),

("DISCONTINUED","DISCONTINUED"))

Groups\_Type\_Choices = (("Arts","Arts"),

("Science","Science"))

LOGIC.PY

from student.models import \*

from student.libraries import \*

def sorted\_set(class\_name,variable\_name):

Total\_Groups = set()

Total\_Year\_Of\_Admission = set()

Total\_Class = set()

students = Student.objects.all()

groups = Groups.objects.all()

for student in students:

Total\_Year\_Of\_Admission.add(student.Year\_Of\_Admission)

for group in groups:

Total\_Groups.add(group.Group)

Total\_Class.add(group.Class)

if variable\_name == 'Group':

return sorted(Total\_Groups)

elif variable\_name == 'Year\_Of\_Admission':

return sorted(Total\_Year\_Of\_Admission)

elif variable\_name == 'Class':

return sorted(Total\_Class)

def Find\_Month(Month):

mon = {'Jan.': 1,'Feb.': 2,'Mar.': 3, 'Apr.':4, 'May.': 5, 'Jun.' : 6, 'Jul.' : 7,'Aug.':8,'Sept.': 9,

'Oct.': 10, 'Nov.' : 11 , 'Dec.' : 12}

return mon[Month]

def Bargraph(Id,Year,Group,Semester,Course):

rollnumbers = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).values\_list('Roll\_Number').order\_by('Roll\_Number')

workingdays = Student\_Attendance.objects.filter(Teacher\_Id\_Attendance = Id).filter(Course\_Code = Course).filter(Semester = Semester)

len\_Workingdays = set()

data = {

'Roll Number' : [],

'Total\_Present' : [],

'Total\_Absent' : [],

'Percentage' : [],

'Marks' : []

}

for i in workingdays:

len\_Workingdays.add(i.Date\_Time\_Original.date())

for roll in rollnumbers:

present = len(workingdays.filter(Roll\_Number\_Attendance = roll).filter(Attendance\_Date = 'PRESENT'))

if len(workingdays) != 0:

data['Roll Number'].append(roll[0])

data['Total\_Present'].append(present)

data['Total\_Absent'].append(len(workingdays.filter(Roll\_Number\_Attendance = roll).filter(Attendance\_Date = 'ABSENT')))

data['Percentage'].append(round((present\*100)/len(len\_Workingdays),2))

else:

pass

ind = [0]

total\_percentage = {

'below 75%': 0,

'above 75%': 0,

'above 80%': 0,

'above 85%': 0,

'above 90%': 0,

}

for i in data['Percentage']:

if i < float(75):

total\_percentage['below 75%'] += 1

data['Marks'].append(0)

print('hello')

elif i < float(80):

total\_percentage['above 75%'] += 1

data['Marks'].append(2)

elif i < float(85):

total\_percentage['above 80%'] += 1

data['Marks'].append(3)

elif i < float(90):

total\_percentage['above 85%'] += 1

data['Marks'].append(4)

elif i <= float(100):

total\_percentage['above 90%'] += 1

data['Marks'].append(5)

if len(workingdays) != 0:

dataframe = pd.DataFrame(data)

dataframe\_percentage = pd.DataFrame({'percent' : ['below 75%','above 75%','above 80%','above 85%','above 90%'],

'index' : [total\_percentage['below 75%'],total\_percentage['above 75%'],total\_percentage['above 80%'],total\_percentage['above 85%'],

total\_percentage['above 90%']]})

average\_percentage = dataframe['Percentage'].mean()

return dataframe\_percentage,average\_percentage,len(len\_Workingdays),dataframe

else:

dataframe = {}

dataframe\_percentage = {}

average\_percentage = 0

return dataframe\_percentage,average\_percentage,len(len\_Workingdays),dataframe

#barplot.show()

def marks\_details(exam\_type,marks):

class\_total = 0

for i in marks:

if i.Marks\_Alloted != None:

class\_total += i.Marks\_Alloted

class\_average = class\_total/len(marks)

return class\_average

def Total\_Marks(marks,roll):

I\_1 = marks.filter(Marks\_Type = 'I Internal')

I\_2 = marks.filter(Marks\_Type = 'II Internal')

Sem = marks.filter(Marks\_Type = 'Semester')

Atten = marks.filter(Marks\_Type = 'Attendance')

Assign = marks.filter(Marks\_Type = 'Assignment')

data = {

'Name' : [],

'Roll' : [],

'I\_1' : [],

'I\_2' : [],

'Assign' : [],

'Sem' : [],

'Atten' : [],

'In\_Total' : [],

'Total' : []

}

for i in roll:

data['Name'].append(i.First\_Name)

data['Roll'].append(i.Roll\_Number)

I\_1\_1 = I\_1.filter(Roll\_Number\_Marks = i)

for j in I\_1\_1:

m1 = j.Marks\_Alloted

data['I\_1'].append(m1)

I\_2\_2 = I\_2.filter(Roll\_Number\_Marks = i)

for j in I\_2\_2:

m2 = j.Marks\_Alloted

data['I\_2'].append(m2)

Assign\_2 = Assign.filter(Roll\_Number\_Marks = i)

for j in Assign\_2:

m3 = j.Marks\_Alloted

data['Assign'].append(m3)

Atten\_3 = Atten.filter(Roll\_Number\_Marks = i)

for j in Atten\_3:

m4 = j.Marks\_Alloted

data['Atten'].append(m4)

Sem\_4 = Sem.filter(Roll\_Number\_Marks = i)

for j in Sem\_4:

m5 = j.Marks\_Alloted

data['Sem'].append(m5)

list\_1 = ['m1','m2','m3','m4','m5']

data\_1 = {'m1':m1,'m2':m2,'m3':m3,'m4':m4,'m5':m5}

for i in list\_1:

if data\_1[i] is None:

data\_1[i] = 0

else:

data\_1[i] = float(data\_1[i])

m6 = ((data\_1['m1']+data\_1['m2'])/4)+data\_1['m3']+data\_1['m4']

data['In\_Total'].append(m6)

data['Total'].append(m6+data\_1['m5'])

dataframe = []

for i in range(len(roll)):

dataframe.append([data['Name'][i],data['Roll'][i],data['I\_1'][i],data['I\_2'][i],data['Assign'][i]

,data['Sem'][i],data['Atten'][i],data['In\_Total'][i],data['Total'][i]])

return dataframe

URLS.PY

from django.urls import path

from student.views import \*

from student.libraries import \*

app\_name= 'student'

urlpatterns = [

path('',Selection, name = 'Selection'),

path('FACULTY/LOGIN/',Login\_Faculty,name = 'Login\_Faculty'),

path('TEACHING/FACULTY/LOGIN/',Login\_Teacher,name = 'Login\_Teacher'),

path('STUDENT/LOGIN/',Login\_Student,name = 'Login\_Student'),

path('Faculty/Homepage/Welcome/',faculty\_first,name = 'faculty\_first'),

path('Student/Homepage/Welcome/',Student\_first,name = 'Student\_first'),

path('Teacher/Homepage/Welcome/',Teacher\_first,name = 'Teacher\_first'),

path('Faculty/<Year>/<Group\_Given>/',faculty\_second, name ='faculty\_second'),

path('Student/Attendance/<Roll\_Number>/<Group>/',Student\_second, name = 'Student\_second'),

path('Teacher/<Id>/<Group>/<Year>/<Semester>/<Course>/',Teacher\_second,name = 'Teacher\_second'),

path('Faculty/<Year>/<First\_Name>/<Roll\_Number>/',faculty\_third, name = 'faculty\_third'),

path('Student/<Roll\_Number>/Attendance/Semester/1/<Group>/',Student\_second, name = 'Student\_second'),

path('Faculty/AddStudent/Details/Register/NewStudent/',Add\_Student, name = 'Add\_Student'),

path('Student/Attendance/<Roll\_Number>/<Group>/<Id>/<Course>/<Semester>/',Student\_eighth, name = 'Student\_eighth'),

path('Teacher/Attendance/<Id>/<Year>/<Group>/<Semester>/<Course>/',Teacher\_third, name = 'Teacher\_third'),

path('Teacher/Marks/<Id>/<Year>/<Group>/<Semester>/<Course>/',Teacher\_eight, name = 'Teacher\_eight'),

path('Student/<Roll\_Number>/Attendance/Semester/2/<Group>/',Student\_third, name = 'Student\_third'),

path('Student/ExaminationMarks/<Roll\_Number>/<Group>/',Student\_ninth, name = 'Student\_ninth'),

path('Teacher/MarkAttendance/<Id>/<Year>/<Group>/<Semester>/<Course>/',Teacher\_fourth, name = 'Teacher\_fourth'),

path('Teacher/IInternal/<Id>/<Year>/<Group>/<Semester>/<Course>/',Teacher\_ninth, name = 'Teacher\_ninth'),

path('Faculty/Edit/<Year>/<Group>/<Roll\_Number>/',Edit\_Student, name = 'Edit\_Student'),

path("Student/Marks/Examination/<Roll\_Number>/<Group>/<Semester>/<Exam>/",Student\_tenth,name = 'Student\_tenth'),

path('Student/<Roll\_Number>/Attendance/Semester/3/<Group>/',Student\_fourth, name = 'Student\_fourth'),

path('Teacher/IIInternal/<Id>/<Year>/<Group>/<Semester>/<Course>/',Teacher\_tenth, name = 'Teacher\_tenth'),

path('Teacher/UpdateAttendance/<Id>/<Year>/<Group>/<Semester>/<Course>/<Date>/',Teacher\_fifth, name = 'Teacher\_fifth'),

path('Teacher/Assignment/<Id>/<Year>/<Group>/<Semester>/<Course>/',Teacher\_eleven, name = 'Teacher\_eleven'),

path('Student/<Roll\_Number>/Attendance/Semester/4/<Group>/',Student\_fifth, name = 'Student\_fifth'),

path('Faculty/Delete/<Year>/<Group>/<Roll\_Number>/',Delete\_Student, name = 'Delete\_Student'),

path('Teacher/Semester/<Id>/<Year>/<Group>/<Semester>/<Course>/',Teacher\_twelve, name = 'Teacher\_twelve'),

path('Faculty/Certificates/<Year>/<Group>/<Roll\_Number>/',Certificate\_Student, name = 'Certificate\_Student'),

path('Teacher/AddAttendance/<Id>/<Year>/<Group>/<Semester>/<Course>/',Teacher\_sixth, name = 'Teacher\_sixth'),

path('Student/<Roll\_Number>/Attendance/Semester/5/<Group>/',Student\_sixth, name = 'Student\_sixth'),

path('Teacher/IInternalSubmit/<Id>/<Year>/<Group>/<Semester>/<Course>/',Teacher\_thirteen, name = 'Teacher\_thirteen'),

path('Teacher/ChangeAttendance/<Id>/<Year>/<Group>/<Semester>/<Course>/<Date>/',Teacher\_seventh, name = 'Teacher\_seventh'),

path('Teacher/IIInternalSubmit/<Id>/<Year>/<Group>/<Semester>/<Course>/',Teacher\_fourteen, name = 'Teacher\_fourteen'),

path('Teacher/AssignmentSubmit/<Id>/<Year>/<Group>/<Semester>/<Course>/',Teacher\_fifteen, name = 'Teacher\_fifteen'),

path('Student/<Roll\_Number>/Attendance/Semester/6/<Group>/',Student\_seventh, name = 'Student\_seventh'),

path('Teacher/SemesterSubmit/<Id>/<Year>/<Group>/<Semester>/<Course>/',Teacher\_sixteen, name = 'Teacher\_sixteen'),

path('Teacher/SubmitAttendance/<Id>/<Year>/<Group>/<Semester>/<Course>/',Teacher\_seventeen, name = 'Teacher\_seventeen'),

]

VALIDATIONS.PY

from django.core.exceptions import ValidationError

from django.utils.translation import gettext\_lazy as \_

from student.libraries import \*

def Phone\_Number(value):

if len(value)!= 10:

raise ValidationError(

\_('The Mobile Number is invalid'),

)

def Image\_Size(Input):

ImageSize = Input.file.size

limit\_Kb\_Min = 20

limit\_Kb\_Max = 300

if ImageSize > limit\_Kb\_Max \* 1024:

raise ValidationError(

\_('The maximum image size is 300 kb'))

VIEW.PY

from django.shortcuts import render,get\_object\_or\_404,HttpResponse

from student.models import \*

from student.logic import \*

from student.libraries import \*

from student.modelforms import \*

# Create your views here.

def Selection(request):

return render(request,'temp/selection.html')

def Login\_Faculty(request):

return render(request,'temp/Facultylogin.html')

def Login\_Teacher(request):

return render(request,'temp/Teacherlogin.html')

def Login\_Student(request):

return render(request,'temp/Studentlogin.html')

def faculty\_first(request):

if request.method == 'POST':

if request.POST.get('Username') == 'Faculty' and request.POST.get('Password') == 'Siddhartha':

Groups\_list = sorted\_set(Student,'Group')

Class\_list = sorted\_set(Student,'Class')

Year\_Of\_Admission\_list = sorted\_set(Student,'Year\_Of\_Admission')

Student\_list = Student.objects.all()

content = {

'Class\_list':Class\_list,

'Groups\_list':Groups\_list,

'Year\_Of\_Admission\_list': Year\_Of\_Admission\_list,

'Student\_list':Student\_list

}

return render(request, 'temp/course.html',content)

else:

return HttpResponse('INVALID')

def faculty\_second(request,Year,Group\_Given):

Student\_Filter\_Year\_Of\_Admission = Student.objects.filter(Year\_Of\_Admission = Year, Group = Group\_Given).order\_by('-Group','Year\_Of\_Admission').order\_by('Roll\_Number')

content = {

'Student\_Filter' : Student\_Filter\_Year\_Of\_Admission

}

return render(request, 'temp/class.html',content)

def faculty\_third(request,Year,First\_Name,Roll\_Number):

Student\_Details = Student.objects.filter(Roll\_Number = Roll\_Number)

content = {

'Student\_Details':Student\_Details

}

return render(request,'temp/student.html',content)

def Add\_Student(request):

if request.method == 'POST':

if Student.objects.filter(Roll\_Number = request.POST.get('Roll\_Number')).exists():

try:

for key, value in request.POST.items():

#print(key,value)

if key == "Roll\_Number":

student = get\_object\_or\_404(Student,Roll\_Number = value)

Id\_Number = value

Student\_Details = Student.objects.get(Roll\_Number = Id\_Number)

form = Student\_ModelForm(request.POST or None, request.FILES or None, instance=Student\_Details)

if form.is\_valid():

edit = form.save(commit = False)

edit.save()

Student\_Details = Student.objects.filter(Roll\_Number = Id\_Number)

print(Student\_Details)

content = {

'Student\_Details':Student\_Details

}

return render(request,'temp/student.html',content)

except:

return HttpResponse("Details Incorrect")

else:

content = {}

form = Student\_ModelForm(request.POST, request.FILES)

print(form)

if form.is\_valid():

form.save()

content = {

'form':form

}

return render(request,"temp/addstudent.html",content)

else:

form = Student\_ModelForm()

content = {

'form':form

}

return render(request,"temp/addstudent.html",content)

def Edit\_Student(request,Year,Group,Roll\_Number):

if request.method == 'POST':

student = get\_object\_or\_404(Roll\_Number = Roll\_Number)

content = {}

form = Student\_ModelForm(request.POST, request.FILES, instance= student)

if form.is\_valid():

form.save()

content = {

'form':form

}

return render("temp/editstudent.html",content)

else:

student = get\_object\_or\_404(Student, Roll\_Number = Roll\_Number)

form = Student\_ModelForm(instance = student)

details = Student.objects.filter(Roll\_Number = Roll\_Number)

content = {

'form':form,

'details':details

}

return render(request,"temp/editstudent.html",content)

def Delete\_Student(request,Year,Group,Roll\_Number):

Student\_Details = Student.objects.get(Roll\_Number = Roll\_Number)

Student\_Details.delete()

Student\_Filter\_Year\_Of\_Admission = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).order\_by('-Group','Year\_Of\_Admission')

content = {

'Student\_Filter' : Student\_Filter\_Year\_Of\_Admission

}

return render(request, 'temp/class.html',content)

def Certificate\_Student(request,Year,Group,Roll\_Number):

Student\_Details = Student.objects.filter(Roll\_Number = Roll\_Number)

content = {

'Details': Student\_Details

}

return render(request,'temp/certificates.html',content)

def Teacher\_first(request):

if request.method == 'POST':

teacher = Teacher.objects.filter(Teacher\_Id = request.POST.get('Username'))

if teacher.exists() and teacher[0].valid\_User\_Password\_Teacher == request.POST.get('Password'):

visit = Teacher\_Course.objects.filter(Teacher\_Id = request.POST.get('Username'))

content = {

'visit' : visit

}

return render(request,'temp/Teacherfirst.html',content)

else:

return HttpResponse('INVALID DETAILS OR NOT A STUDENT')

def Teacher\_second(request,Id,Group,Year,Semester,Course):

Class\_Students = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number')

content = {

'Id' : Id,

'class\_student' : Class\_Students,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request, 'temp/Teachersecond.html',content)

def Teacher\_third(request,Id,Year,Group,Semester,Course):

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).values\_list('Roll\_Number')

Previous = Student\_Attendance.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Roll\_Number\_Attendance\_\_in = Rolls).order\_by('Date\_Time\_Original')

Previous\_Date = Previous.values\_list('Date\_Time\_Original')

Total\_strength = len(Rolls)

Dates = {}

for p in Previous:

Date = p.Date\_Time\_Original.date()

Dates[Date] = len(Previous.filter(Date\_Time\_Original = p.Date\_Time\_Original).filter(Attendance\_Date = 'PRESENT'))

content = {

'Id' : Id,

'T' : Total\_strength,

'Dates' : Dates,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request, 'temp/Teacherthird.html',content)

def Teacher\_fourth(request,Id,Year,Group,Semester,Course):

if request.POST['SUBMIT'] == 'Mark':

Attendance = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number')

Roll =Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number').values\_list('Roll\_Number')

Rolls = json.dumps(list(Roll))

content = {

'Id' : Id,

'Attendance':Attendance,

'Roll':Rolls,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request,'temp/Teacherfourth.html',content)

elif request.POST['SUBMIT'] == 'Delete':

print(request.POST)

values = list(request.POST.keys())

for i in range(2,len(values)):

if request.POST[values[i]] == 'on':

Date = values[i].split(" ")

Day = int(Date[1].split(",")[0])

#print(Date[9:13],Date[5:7])

Month = Find\_Month(Date[0])

year = int(Date[2])

Values = Student\_Attendance.objects.filter(Teacher\_Id\_Attendance = Id).filter(Course\_Code = Course).filter(Semester = Semester).filter(Date\_Time\_Original\_\_date = datetime.date(year,Month,Day))

for i in Values:

#print(i)

i.delete()

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).values\_list('Roll\_Number')

Previous = Student\_Attendance.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Roll\_Number\_Attendance\_\_in = Rolls).order\_by('Date\_Time\_Original')

Previous\_Date = Previous.values\_list('Date\_Time\_Original')

Total\_strength = len(Rolls)

Dates = {}

for p in Previous:

Date = p.Date\_Time\_Original.date()

Dates[Date] = len(Previous.filter(Date\_Time\_Original = p.Date\_Time\_Original).filter(Attendance\_Date = 'PRESENT'))

content = {

'Id' : Id,

'T' : Total\_strength,

'Dates' : Dates,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request, 'temp/Teacherthird.html',content)

elif request.POST['SUBMIT'] == 'Report':

studentgraph = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number')

dataframe = Bargraph(Id,Year,Group,Semester,Course)[0]

average\_present = round(Bargraph(Id,Year,Group,Semester,Course)[1],2)

average\_absent = float(100)-average\_present

cos = Courses.objects.filter(Course\_Code = Course)

dataframe2 = Bargraph(Id,Year,Group,Semester,Course)[3]

datas = list()

for i in range(len(studentgraph)):

if len(dataframe2) != 0:

datas.append(list(dataframe2.loc[i]))

else:

pass

if len(dataframe2) != 0:

content = {

'Id' : Id,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course,

'student' : studentgraph,

'dataframe' : datas,

'labels' : ['Average Present','Average Absent'],

'data' : [average\_present,average\_absent],

'barlabels' : list(dataframe['percent']),

'bardata' : list(dataframe['index']),

'class' : Group,

'Sem' : Semester,

'course' : Course,

'Cos' : cos[0],

'T\_W' : Bargraph(Id,Year,Group,Semester,Course)[2],

'len' : len(studentgraph),

'p' : round(Bargraph(Id,Year,Group,Semester,Course)[1],2),

'below' : list(dataframe['index'])[0]

}

return render(request, 'temp/TeacherReport.html',content)

else:

return HttpResponse("No Attendance to Generate The Report")

def Teacher\_fifth(request,Id,Year,Group,Semester,Course,Date):

D = Date.split(" ")

Day = int(D[1].split(",")[0])

Month = Find\_Month(D[0])

year = int(D[2])

Values = Student\_Attendance.objects.filter(Teacher\_Id\_Attendance = Id).filter(Course\_Code = Course).filter(Semester = Semester).filter(Date\_Time\_Original\_\_date = datetime.date(year,Month,Day)).order\_by('Date\_Time\_Original')

content = {

'Date' : Date[5:7],

'Month' : Date[0:3],

'Year' : Date[9:13],

'Date\_Total' : Date,

'Attendance' : Values,

'Id' : Id,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request, 'temp/Teacherfifth.html',content)

def Teacher\_sixth(request,Id,Year,Group,Semester,Course):

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).values\_list('Roll\_Number')

Previous = Student\_Attendance.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Roll\_Number\_Attendance\_\_in = Rolls).order\_by('Date\_Time\_Original')

Previous\_Date = set()

Course\_original = Course

Id\_original = Id

for i in Previous.values\_list('Date\_Time\_Original'):

Previous\_Date.add(str(i[0].date()))

value = list(request.POST)

li = value[3:len(value)]

Attendance = {}

for i in range(0,len(li)):

Attendance[li[i]] = request.POST.get(li[i])

Course = Courses.objects.get(Course\_Code = Course)

Id = Teacher.objects.get(Teacher\_Id = Id)

for key in Attendance:

Key = Student.objects.get(Roll\_Number = key)

if request.POST.get('Date') not in Previous\_Date:

Instance = Student\_Attendance(Date\_Time\_Original = request.POST.get('Date'),Semester = Semester,

Course\_Code = Course,Roll\_Number\_Attendance = Key, Teacher\_Id\_Attendance = Id,

Attendance\_Date = Attendance[key])

Instance.save()

else:

return HttpResponse("The Attendance of the Date exists?Are You Trying To Update The Attendance?")

Total\_strength = len(Rolls)

Dates = {}

for p in Previous:

Date = p.Date\_Time\_Original.date()

Dates[Date] = len(Previous.filter(Date\_Time\_Original = p.Date\_Time\_Original).filter(Attendance\_Date = 'PRESENT'))

content = {

'Id' : Id\_original,

'T' : Total\_strength,

'Dates' : Dates,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course\_original

}

return render(request, 'temp/Teacherthird.html', content)

def Teacher\_seventh(request,Id,Year,Group,Semester,Course,Date):

D = Date.split(" ")

Day = int(D[1].split(",")[0])

Month = Find\_Month(D[0])

year = int(D[2])

Values = Student\_Attendance.objects.filter(Teacher\_Id\_Attendance = Id).filter(Course\_Code = Course).filter(Semester = Semester).filter(Date\_Time\_Original\_\_date = datetime.date(year,Month,Day)).order\_by('Date\_Time\_Original')

value = list(request.POST)

li = value[3:len(value)]

Attendance = {}

for i in range(0,len(li)):

Attendance[li[i]] = request.POST.get(li[i])

for i in range(len(Values)):

if list(Attendance.keys())[i] == Values[i].Roll\_Number\_Attendance.Roll\_Number and Attendance[list(Attendance.keys())[i]] != Values[i].Attendance\_Date:

instance = Values[i]

instance.Attendance\_Date = Attendance[list(Attendance.keys())[i]]

instance.save()

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).values\_list('Roll\_Number')

Previous = Student\_Attendance.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Roll\_Number\_Attendance\_\_in = Rolls).order\_by('Date\_Time\_Original')

Previous\_Date = Previous.values\_list('Date\_Time\_Original')

Total\_strength = len(Rolls)

Dates = {}

for p in Previous:

Date = p.Date\_Time\_Original.date()

Dates[Date] = len(Previous.filter(Date\_Time\_Original = p.Date\_Time\_Original).filter(Attendance\_Date = 'PRESENT'))

content = {

'Id' : Id,

'T' : Total\_strength,

'Dates' : Dates,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request, 'temp/Teacherthird.html',content)

def Teacher\_eight(request,Id,Year,Group,Semester,Course):

Student\_List = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number')

Types = {'I INTERNAL','II INTERNAL','ASSIGNMENT','SEMESTER'}

content = {

'Attendance' : Student\_List,

'Types' : Types,

'Id' : Id,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request,'temp/Teachereight.html',content)

def Teacher\_ninth(request,Id,Year,Group,Semester,Course):

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).values\_list('Roll\_Number')

Internal\_1 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Marks\_Type = 'I Internal').filter(Roll\_Number\_Marks\_\_in = Rolls)

Total\_Strength = len(Internal\_1)

Absent = Internal\_1.filter(Marks\_Alloted = None)

Total\_Absent = len(Absent)

Total\_Present = Total\_Strength - Total\_Absent

Total\_Fail = 0

Total\_Pass = 0

for i in Internal\_1:

if i.Marks\_Alloted == None or i.Marks\_Alloted < 12:

Total\_Fail += 1

else:

Total\_Pass += 1

if len(Internal\_1) != 0:

content = {

'T\_S' : len(Rolls),

'average': marks\_details('I Internal',Internal\_1),

'labels' :['Total Students Passed','Total Students Failed'],

'data' : [Total\_Pass,Total\_Fail],

'Attendance' : Internal\_1,

'Absent' : Total\_Absent,

'Present' : Total\_Present,

'Id' : Id,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course,

}

return render(request, 'temp/Teacherninehalf.html',content)

else:

Internal\_2 = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number')

content = {

'Attendance' : Internal\_2,

'Id' : Id,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request, 'temp/Teacherninth.html',content)

def Teacher\_tenth(request,Id,Year,Group,Semester,Course):

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).values\_list('Roll\_Number')

Internal\_2 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Marks\_Type = 'II Internal').filter(Roll\_Number\_Marks\_\_in = Rolls)

Total\_Strength = len(Internal\_2)

Absent = Internal\_2.filter(Marks\_Alloted = None)

Total\_Absent = len(Absent)

Total\_Present = Total\_Strength - Total\_Absent

Total\_Fail = 0

Total\_Pass = 0

for i in Internal\_2:

if i.Marks\_Alloted == None or i.Marks\_Alloted < 12:

Total\_Fail += 1

else:

Total\_Pass += 1

if len(Internal\_2) != 0:

content = {

'T\_S' : len(Rolls),

'average': marks\_details('II Internal',Internal\_2),

'labels' :['Total Students Passed','Total Students Failed'],

'data' : [Total\_Pass,Total\_Fail],

'Attendance' : Internal\_2,

'Id' : Id,

'Absent' : Total\_Absent,

'Present' : Total\_Present,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request, 'temp/Teachertenhalf.html',content)

else:

Internal\_2 = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number')

content = {

'Attendance' : Internal\_2,

'Id' : Id,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request, 'temp/Teachertenth.html',content)

def Teacher\_eleven(request,Id,Year,Group,Semester,Course):

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).values\_list('Roll\_Number')

Assignment = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Marks\_Type = 'Assignment').filter(Roll\_Number\_Marks\_\_in = Rolls)

if len(Assignment) != 0:

content = {

'Attendance' : Assignment,

'Id' : Id,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request, 'temp/Teacherelevenhalf.html',content)

else:

Assignment = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number')

content = {

'Attendance' : Assignment,

'Id' : Id,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request, 'temp/Teachereleven.html',content)

def Teacher\_twelve(request,Id,Year,Group,Semester,Course):

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).values\_list('Roll\_Number')

Semesters = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Marks\_Type = 'Semester').filter(Roll\_Number\_Marks\_\_in = Rolls)

Total\_Strength = len(Semesters)

Absent = Semesters.filter(Marks\_Alloted = None)

Total\_Absent = len(Absent)

Total\_Present = Total\_Strength - Total\_Absent

Total\_Fail = 0

Total\_Pass = 0

for i in Semesters:

if i.Marks\_Alloted == None or i.Marks\_Alloted < 35:

Total\_Fail += 1

else:

Total\_Pass += 1

if len(Semesters) != 0:

content = {

'T\_S' : len(Rolls),

'average': marks\_details('Semester',Semesters),

'labels' :['Total Students Passed','Total Students Failed'],

'data' : [Total\_Pass,Total\_Fail],

'Attendance' : Semesters,

'Absent' : Total\_Absent,

'Present' : Total\_Present,

'Id' : Id,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request, 'temp/Teachertwelvehalf.html',content)

else:

Semesters = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number')

content = {

'Attendance' : Semesters,

'Id' : Id,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course

}

return render(request, 'temp/Teachertwelve.html',content)

def Teacher\_thirteen(request,Id,Year,Group,Semester,Course):

Cours = Course

Ids = Id

A = request.POST

B = list(A.keys())[3:len(A)]

Course = Courses.objects.get(Course\_Code = Course)

Id = Teacher.objects.get(Teacher\_Id = Id)

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).values\_list('Roll\_Number')

Previous\_Internal\_1 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls)

previous\_marks\_type = set()

for i in Previous\_Internal\_1.values\_list('Marks\_Type'):

previous\_marks\_type.add(i[0])

if request.POST['SUBMIT'] == 'submit' and "I Internal" not in previous\_marks\_type:

for i in B:

if len(A[i]) != 0:

Instance = Student\_Marks(Semester = Semester,Marks\_Type = 'I Internal',Course\_Code = Course,Roll\_Number\_Marks = Student.objects.get(Roll\_Number= i),Teacher\_Id\_Marks = Id,Marks\_Total = A['Total Marks'],Marks\_Alloted = float(A[i]))

Instance.save()

else:

Instance = Student\_Marks(Semester = Semester,Marks\_Type = 'I Internal',Course\_Code = Course,Roll\_Number\_Marks = Student.objects.get(Roll\_Number= i),Teacher\_Id\_Marks = Id,Marks\_Total = A['Total Marks'])

Instance.save()

Previous\_Internal\_1 = Previous\_Internal\_1.filter(Marks\_Type = 'I Internal')

Previous\_1 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'I Internal')

Total\_Strength = len(Previous\_1)

Absent = Previous\_1.filter(Marks\_Alloted = None)

Total\_Absent = len(Absent)

Total\_Present = Total\_Strength - Total\_Absent

Total\_Fail = 0

Total\_Pass = 0

for i in Previous\_1:

if i.Marks\_Alloted == None or i.Marks\_Alloted < 12:

Total\_Fail += 1

else:

Total\_Pass += 1

content = {

'T\_S' : len(Rolls),

'average': marks\_details('I Internal',Previous\_1),

'labels' :['Total Students Passed','Total Students Failed'],

'data' : [Total\_Pass,Total\_Fail],

'Attendance' : Previous\_Internal\_1,

'Absent' : Total\_Absent,

'Present': Total\_Present,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request,'temp/Teacherninehalf.html',content)

elif request.POST['SUBMIT'] == 'updatepage':

Internal\_1 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Marks\_Type = 'I Internal').filter(Roll\_Number\_Marks\_\_in = Rolls)

content = {

'Attendance' : Internal\_1,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request, 'temp/Teacherninehalfupdate.html',content)

elif request.POST['SUBMIT'] == 'updatemarks':

Previous\_Internal\_1 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'I Internal')

for student in Previous\_Internal\_1:

for s in B:

if student.Roll\_Number\_Marks.Roll\_Number == s and str(student.Marks\_Alloted) != A[s] and len(A[s]) != 0:

student.Marks\_Alloted = float(A[s])

student.save()

elif student.Roll\_Number\_Marks.Roll\_Number == s and str(student.Marks\_Alloted) != A[s] and len(A[s]) == 0:

student.Marks\_Alloted = None

student.save()

Previous\_1 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'I Internal')

Total\_Strength = len(Previous\_1)

Absent = Previous\_1.filter(Marks\_Alloted = None)

Total\_Absent = len(Absent)

Total\_Present = Total\_Strength - Total\_Absent

Total\_Fail = 0

Total\_Pass = 0

for i in Previous\_1:

if i.Marks\_Alloted == None or i.Marks\_Alloted < 12:

Total\_Fail += 1

else:

Total\_Pass += 1

content = {

'Absent' : Total\_Absent,

'Present': Total\_Present,

'T\_S' : len(Rolls),

'average': marks\_details('I Internal',Previous\_1),

'labels' :['Total Students Passed','Total Students Failed'],

'data' : [Total\_Pass,Total\_Fail],

'Attendance' : Previous\_1,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request,'temp/Teacherninehalf.html',content)

elif request.POST['SUBMIT'] == 'delete':

Previous\_1 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'I Internal')

for i in Previous\_1:

i.delete()

previous\_marks = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number')

content = {

'Attendance' : previous\_marks,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request,'temp/Teacherninth.html',content)

def Teacher\_fourteen(request,Id,Year,Group,Semester,Course):

Cours = Course

Ids = Id

A = request.POST

B = list(A.keys())[3:len(A)]

Course = Courses.objects.get(Course\_Code = Course)

Id = Teacher.objects.get(Teacher\_Id = Id)

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).values\_list('Roll\_Number')

Previous\_Internal\_2 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls)

previous\_marks\_type = set()

for i in Previous\_Internal\_2.values\_list('Marks\_Type'):

previous\_marks\_type.add(i[0])

if request.POST['SUBMIT'] == 'submit' and "II Internal" not in previous\_marks\_type:

for i in B:

if len(A[i]) != 0:

Instance = Student\_Marks(Semester = Semester,Marks\_Type = 'II Internal',Course\_Code = Course,Roll\_Number\_Marks = Student.objects.get(Roll\_Number= i),Teacher\_Id\_Marks = Id,Marks\_Total = A['Total Marks'],Marks\_Alloted = float(A[i]))

Instance.save()

else:

Instance = Student\_Marks(Semester = Semester,Marks\_Type = 'II Internal',Course\_Code = Course,Roll\_Number\_Marks = Student.objects.get(Roll\_Number= i),Teacher\_Id\_Marks = Id,Marks\_Total = A['Total Marks'])

Instance.save()

Previous\_Internal\_1 = Previous\_Internal\_2.filter(Marks\_Type = 'II Internal')

Previous\_1 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'II Internal')

Total\_Strength = len(Previous\_1)

Absent = Previous\_1.filter(Marks\_Alloted = None)

Total\_Absent = len(Absent)

Total\_Present = Total\_Strength - Total\_Absent

Total\_Fail = 0

Total\_Pass = 0

for i in Previous\_1:

if i.Marks\_Alloted == None or i.Marks\_Alloted < 12:

Total\_Fail += 1

else:

Total\_Pass += 1

content = {

'T\_S' : len(Rolls),

'average': marks\_details('II Internal',Previous\_1),

'labels' :['Total Students Passed','Total Students Failed'],

'data' : [Total\_Pass,Total\_Fail],

'Attendance' : Previous\_Internal\_1,

'Absent' : Total\_Absent,

'Present': Total\_Present,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request,'temp/Teachertenhalf.html',content)

elif request.POST['SUBMIT'] == 'updatepage':

Internal\_2 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Marks\_Type = 'II Internal').filter(Roll\_Number\_Marks\_\_in = Rolls)

content = {

'Attendance' : Internal\_2,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request, 'temp/Teachertenhalfupdate.html',content)

elif request.POST['SUBMIT'] == 'updatemarks':

Previous\_Internal\_2 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'II Internal')

for student in Previous\_Internal\_2:

for s in B:

if student.Roll\_Number\_Marks.Roll\_Number == s and str(student.Marks\_Alloted) != A[s] and len(A[s]) != 0:

student.Marks\_Alloted = float(A[s])

student.save()

elif student.Roll\_Number\_Marks.Roll\_Number == s and str(student.Marks\_Alloted) != A[s] and len(A[s]) == 0:

student.Marks\_Alloted = None

student.save()

Previous\_1 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'II Internal')

Total\_Strength = len(Previous\_1)

Absent = Previous\_1.filter(Marks\_Alloted = None)

Total\_Absent = len(Absent)

Total\_Present = Total\_Strength - Total\_Absent

Total\_Fail = 0

Total\_Pass = 0

for i in Previous\_1:

if i.Marks\_Alloted == None or i.Marks\_Alloted < 12:

Total\_Fail += 1

else:

Total\_Pass += 1

content = {

'Absent' : Total\_Absent,

'Present': Total\_Present,

'T\_S' : len(Rolls),

'average': marks\_details('II Internal',Previous\_1),

'labels' :['Total Students Passed','Total Students Failed'],

'data' : [Total\_Pass,Total\_Fail],

'Attendance' : Previous\_1,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request,'temp/Teachertenhalf.html',content)

elif request.POST['SUBMIT'] == 'delete':

Previous\_2 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = "II Internal")

for i in Previous\_2:

i.delete()

previous\_marks = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number')

content = {

'Attendance' : previous\_marks,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request,'temp/Teachertenth.html',content)

def Teacher\_fifteen(request,Id,Year,Group,Semester,Course):

Cours = Course

Ids = Id

A = request.POST

B = list(A.keys())[3:len(A)]

Course = Courses.objects.get(Course\_Code = Course)

Id = Teacher.objects.get(Teacher\_Id = Id)

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).values\_list('Roll\_Number')

Previous\_Assignment = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls)

previous\_marks\_type = set()

for i in Previous\_Assignment.values\_list('Marks\_Type'):

previous\_marks\_type.add(i[0])

if request.POST['SUBMIT'] == 'submit' and "Assignment" not in previous\_marks\_type:

for i in B:

Instance = Student\_Marks(Semester = Semester,Marks\_Type = 'Assignment',Course\_Code = Course,Roll\_Number\_Marks = Student.objects.get(Roll\_Number= i),Teacher\_Id\_Marks = Id,Marks\_Total = A['Total Marks'],Marks\_Alloted = float(A[i]))

Instance.save()

Previous\_Assignment = Previous\_Assignment.filter(Marks\_Type = 'Assignment')

content = {

'Attendance' : Previous\_Assignment,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request,'temp/Teacherelevenhalf.html',content)

elif request.POST['SUBMIT'] == 'updatepage':

Assignment = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Marks\_Type = 'Assignment').filter(Roll\_Number\_Marks\_\_in = Rolls)

content = {

'Attendance' : Assignment,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request, 'temp/Teacherelevenhalfupdate.html',content)

elif request.POST['SUBMIT'] == 'updatemarks':

Previous\_Assignment = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'Assignment')

for student in Previous\_Assignment:

for s in B:

if student.Roll\_Number\_Marks.Roll\_Number == s and str(student.Marks\_Alloted) != A[s]:

student.Marks\_Alloted = float(A[s])

student.save()

Previous\_Assignment\_2 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'Assignment')

content = {

'Attendance' : Previous\_Assignment\_2,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request,'temp/Teacherelevenhalf.html',content)

elif request.POST['SUBMIT'] == 'delete':

Previous\_Assignment = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'Assignment')

for i in Previous\_Assignment:

i.delete()

previous\_marks = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number')

content = {

'Attendance' : previous\_marks,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request,'temp/Teachereleven.html',content)

def Teacher\_sixteen(request,Id,Year,Group,Semester,Course):

Cours = Course

Ids = Id

A = request.POST

B = list(A.keys())[3:len(A)]

Course = Courses.objects.get(Course\_Code = Course)

Id = Teacher.objects.get(Teacher\_Id = Id)

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).values\_list('Roll\_Number')

Previous\_Semester = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls)

previous\_marks\_type = set()

for i in Previous\_Semester.values\_list('Marks\_Type'):

previous\_marks\_type.add(i[0])

if request.POST['SUBMIT'] == 'submit' and "Semester" not in previous\_marks\_type:

for i in B:

if len(A[i]) != 0:

Instance = Student\_Marks(Semester = Semester,Marks\_Type = 'Semester',Course\_Code = Course,Roll\_Number\_Marks = Student.objects.get(Roll\_Number= i),Teacher\_Id\_Marks = Id,Marks\_Total = A['Total Marks'],Marks\_Alloted = float(A[i]))

Instance.save()

else:

Instance = Student\_Marks(Semester = Semester,Marks\_Type = 'Semester',Course\_Code = Course,Roll\_Number\_Marks = Student.objects.get(Roll\_Number= i),Teacher\_Id\_Marks = Id,Marks\_Total = A['Total Marks'])

Instance.save()

Previous\_Internal\_1 = Previous\_Semester.filter(Marks\_Type = 'Semester')

Previous\_1 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'Semester')

Total\_Strength = len(Previous\_1)

Absent = Previous\_1.filter(Marks\_Alloted = None)

Total\_Absent = len(Absent)

Total\_Present = Total\_Strength - Total\_Absent

Total\_Fail = 0

Total\_Pass = 0

for i in Previous\_1:

if i.Marks\_Alloted == None or i.Marks\_Alloted < 35:

Total\_Fail += 1

else:

Total\_Pass += 1

content = {

'T\_S' : len(Rolls),

'average': marks\_details('Semester',Previous\_1),

'labels' :['Total Students Passed','Total Students Failed'],

'data' : [Total\_Pass,Total\_Fail],

'Attendance' : Previous\_Internal\_1,

'Absent' : Total\_Absent,

'Present': Total\_Present,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request,'temp/Teachertwelvehalf.html',content)

elif request.POST['SUBMIT'] == 'updatepage':

Semesters = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Marks\_Type = 'Semester').filter(Roll\_Number\_Marks\_\_in = Rolls)

content = {

'Attendance' : Semesters,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request, 'temp/Teachertwelvehalfupdate.html',content)

elif request.POST['SUBMIT'] == 'updatemarks':

Previous\_Semester = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'Semester')

for student in Previous\_Semester:

for s in B:

if student.Roll\_Number\_Marks.Roll\_Number == s and str(student.Marks\_Alloted) != A[s] and len(A[s]) != 0:

student.Marks\_Alloted = float(A[s])

student.save()

elif student.Roll\_Number\_Marks.Roll\_Number == s and str(student.Marks\_Alloted) != A[s] and len(A[s]) == 0:

student.Marks\_Alloted = None

student.save()

Previous\_1 = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'Semester')

Total\_Strength = len(Previous\_1)

Absent = Previous\_1.filter(Marks\_Alloted = None)

Total\_Absent = len(Absent)

Total\_Present = Total\_Strength - Total\_Absent

Total\_Fail = 0

Total\_Pass = 0

for i in Previous\_1:

if i.Marks\_Alloted == None or i.Marks\_Alloted < 35:

Total\_Fail += 1

else:

Total\_Pass += 1

content = {

'Absent' : Total\_Absent,

'Present': Total\_Present,

'T\_S' : len(Rolls),

'average': marks\_details('Semester',Previous\_1),

'labels' :['Total Students Passed','Total Students Failed'],

'data' : [Total\_Pass,Total\_Fail],

'Attendance' : Previous\_1,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request,'temp/Teachertwelvehalf.html',content)

elif request.POST['SUBMIT'] == 'delete':

Previous\_Semester = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'Semester')

for i in Previous\_Semester:

i.delete()

previous\_marks = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number')

content = {

'Attendance' : previous\_marks,

'Id' : Ids,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Cours

}

return render(request,'temp/Teachertwelve.html',content)

elif request.POST['SUBMIT'] == 'Report':

Previous\_Marks = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls)

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).order\_by('Roll\_Number')

result = Total\_Marks(Previous\_Marks,Rolls)

content = {

'result' : result,

'Attendance' : Previous\_Marks,

'rolls' : Rolls

}

return render(request,'temp/TeacherReport2.html',content)

def Teacher\_seventeen(request,Id,Year,Semester,Course,Group):

A = request.POST

B = list(A.keys())[2:len(A)]

Rolls = Student.objects.filter(Year\_Of\_Admission = Year).filter(Group = Group).values\_list('Roll\_Number')

Previous\_Attendance = Student\_Marks.objects.filter(Semester = Semester).filter(Course\_Code = Course).filter(Teacher\_Id\_Marks = Id).filter(Roll\_Number\_Marks\_\_in = Rolls).filter(Marks\_Type = 'Attendance')

print(Previous\_Attendance)

for i in B:

if len(Previous\_Attendance.filter(Roll\_Number\_Marks = i)) == 0:

instance = Student\_Marks(Semester = Semester,Marks\_Type = 'Attendance',Course\_Code = Courses.objects.get(Course\_Code = Course),Roll\_Number\_Marks = Student.objects.get(Roll\_Number= i),Teacher\_Id\_Marks = Teacher.objects.get(Teacher\_Id = Id),Marks\_Total = 5,Marks\_Alloted = float(A[i]))

instance.save()

else:

for student in Previous\_Attendance:

if student.Roll\_Number\_Marks.Roll\_Number == i and student.Marks\_Alloted != float(A[i]):

student.Marks\_Alloted = float(A[i])

student.save()

studentgraph = Student.objects.filter(Group = Group).filter(Year\_Of\_Admission = Year).order\_by('Roll\_Number')

dataframe = Bargraph(Id,Year,Group,Semester,Course)[0]

average\_present = round(Bargraph(Id,Year,Group,Semester,Course)[1],2)

average\_absent = float(100)-average\_present

cos = Courses.objects.filter(Course\_Code = Course)

dataframe2 = Bargraph(Id,Year,Group,Semester,Course)[3]

datas = list()

for i in range(len(studentgraph)):

datas.append(list(dataframe2.loc[i]))

content = {

'Id' : Id,

'Year' : Year,

'Group' : Group,

'Semester' : Semester,

'Course' : Course,

'student' : studentgraph,

'dataframe' : datas,

'labels' : ['Average Present','Average Absent'],

'data' : [average\_present,average\_absent],

'barlabels' : list(dataframe['percent']),

'bardata' : list(dataframe['index']),

'class' : Group,

'Sem' : Semester,

'course' : Course,

'Cos' : cos[0],

'T\_W' : Bargraph(Id,Year,Group,Semester,Course)[2],

'len' : len(studentgraph),

'p' : round(Bargraph(Id,Year,Group,Semester,Course)[1],2),

'below' : list(dataframe['index'])[0]

}

return render(request, 'temp/TeacherReport.html',content)

def Student\_first(request):

if request.method == 'POST':

student = Student.objects.filter(Roll\_Number = request.POST.get('Username'))

if student.exists() and student[0].valid\_User\_Password == request.POST.get('Password'):

content = {

'student' : student

}

return render(request,'temp/Studentdetails.html',content)

else:

return HttpResponse('INVALID DETAILS OR NOT A STUDENT')

#STUDENT ATTENDANCE

def Student\_second(request,Roll\_Number,Group):

Courses\_Involved = Teacher\_Course.objects.filter(Group = Group).filter(Semester\_Teacher = "1")

if len(Courses\_Involved) != 0:

content = {

#'Attendance' : courses\_attendance,

'Semester' : 'Semester 1',

'Roll\_Number': Roll\_Number,

'Attendance' : Courses\_Involved,

'Group' : Group,

'sem' : "1"

}

return render(request,'temp/Studentsecond.html',content)

else:

content = {

'Semester' : 'Semester 1',

"no": "NO ATTENDANCE POSTED YET",

'Roll\_Number': Roll\_Number,

'Attendance' : Courses\_Involved,

'Group' : Group

}

return render(request,'temp/Studentthird.html',content)

#STUDENT MARKS

def Student\_third(request,Roll\_Number,Group):

Courses\_Involved = Teacher\_Course.objects.filter(Group = Group).filter(Semester\_Teacher = "2")

if len(Courses\_Involved) != 0:

content = {

'Semester' : 'Semester 2',

'Roll\_Number': Roll\_Number,

'Attendance' : Courses\_Involved,

'Group' : Group,

'sem' : "2"

}

return render(request,'temp/Studentsecond.html',content)

else:

content = {

'Semester' : 'Semester 2',

"no": "NO ATTENDANCE POSTED YET",

'Roll\_Number': Roll\_Number,

'Attendance' : Courses\_Involved,

'Group' : Group

}

return render(request,'temp/Studentthird.html',content)

def Student\_fourth(request,Roll\_Number,Group):

Courses\_Involved = Teacher\_Course.objects.filter(Group = Group).filter(Semester\_Teacher = "3")

if len(Courses\_Involved) != 0:

content = {

'Semester' : 'Semester 3',

'Roll\_Number': Roll\_Number,

'Attendance' : Courses\_Involved,

'Group' : Group,

"sem" : "3"

}

return render(request,'temp/Studentsecond.html',content)

else:

content = {

'Semester' : 'Semester 3',

"no": "NO ATTENDANCE POSTED YET",

'Roll\_Number': Roll\_Number,

'Attendance' : Courses\_Involved,

'Group' : Group

}

return render(request,'temp/Studentthird.html',content)

def Student\_fifth(request,Roll\_Number,Group):

Courses\_Involved = Teacher\_Course.objects.filter(Group = Group).filter(Semester\_Teacher = "4")

if len(Courses\_Involved) != 0:

content = {

'Semester' : 'Semester 4',

'Roll\_Number': Roll\_Number,

'Attendance' : Courses\_Involved,

'Group' : Group,

"sem" : "4"

}

return render(request,'temp/Studentsecond.html',content)

else:

content = {

'Semester' : 'Semester 4',

"no": "NO ATTENDANCE POSTED YET",

'Roll\_Number': Roll\_Number,

'Attendance' : Courses\_Involved,

'Group' : Group

}

return render(request,'temp/Studentthird.html',content)

def Student\_sixth(request,Roll\_Number,Group):

Courses\_Involved = Teacher\_Course.objects.filter(Group = Group).filter(Semester\_Teacher = "5")

if len(Courses\_Involved) != 0:

content = {

'Semester' : 'Semester 5',

'Roll\_Number': Roll\_Number,

'Attendance' : Courses\_Involved,

'Group' : Group,

"sem" : "5"

}

return render(request,'temp/Studentsecond.html',content)

else:

content = {

'Semester' : 'Semester 5',

"no": "NO ATTENDANCE POSTED YET",

'Roll\_Number': Roll\_Number,

'Attendance' : Courses\_Involved,

'Group' : Group

}

return render(request,'temp/Studentthird.html',content)

def Student\_seventh(request,Roll\_Number,Group):

Courses\_Involved = Teacher\_Course.objects.filter(Group = Group).filter(Semester\_Teacher = "6")

if len(Courses\_Involved) != 0:

content = {

'Semester' : 'Semester 6',

'Roll\_Number': Roll\_Number,

'Attendance' : Courses\_Involved,

'Group' : Group,

"sem" : "6"

}

return render(request,'temp/Studentsecond.html',content)

else:

content = {

'Semester' : 'Semester 6',

"no": "NO ATTENDANCE POSTED YET",

'Roll\_Number': Roll\_Number,

'Attendance' : Courses\_Involved,

'Group' : Group

}

return render(request,'temp/Studentthird.html',content)

def Student\_eighth(request,Roll\_Number,Group,Id,Course,Semester):

Student\_attendance = Student\_Attendance.objects.filter(Roll\_Number\_Attendance = Roll\_Number,

Teacher\_Id\_Attendance = Id).filter(Course\_Code = Course).filter(Semester = Semester).order\_by('Date\_Time\_Original')

Total\_working = len(Student\_attendance)

Total\_Present = len(Student\_Attendance.objects.filter(Roll\_Number\_Attendance = Roll\_Number,

Teacher\_Id\_Attendance = Id).filter(Course\_Code = Course).filter(Semester = Semester).filter(Attendance\_Date = 'PRESENT'))

Total\_Absent = Total\_working - Total\_Present

if Total\_working != 0:

content = {

'Attendance' : Student\_attendance,

'T\_W' : Total\_working,

'T\_P' : Total\_Present,

'T\_A' : Total\_Absent

}

return render(request,'temp/Studentfourth.html',content)

else:

return HttpResponse('No Attendane posted yet')

def Student\_ninth(request,Roll\_Number,Group):

Semesters = ('SEMESTER I','SEMESTER II','SEMESTER III','SEMESTER IV','SEMESTER V','SEMESTER VI')

Marks = ('I INTERNAL','II INTERNAL','SEMESTER','ASSIGNMENT','ATTENDANCE')

Internal\_1\_marks = Student\_Marks.objects.filter(Roll\_Number\_Marks = Roll\_Number).filter(Marks\_Type = 'I Internal').filter(Semester = "1")

content ={

'Student' : Internal\_1\_marks,

'Roll\_Number' : Roll\_Number,

'Group' : Group,

'Attendance' : Semesters,

'Marks' : Marks

}

return render(request,'temp/Studentfifth.html',content)

def Student\_tenth(request,Roll\_Number,Group,Semester,Exam):

Semesters = ('SEMESTER I','SEMESTER II','SEMESTER III','SEMESTER IV','SEMESTER V','SEMESTER VI')

Semester\_Rank = {'SEMESTER I' : '1','SEMESTER II' : '2','SEMESTER III' : '3',

'SEMESTER IV' : '4','SEMESTER V' : '5','SEMESTER VI' : '6'}

Marks = ('I INTERNAL','II INTERNAL','SEMESTER','ASSIGNMENT','ATTENDANCE')

Marks\_Rank = {'I INTERNAL' : 'I Internal','II INTERNAL':'II Internal','SEMESTER':'Semester','ASSIGNMENT':'Assignment','ATTENDANCE':'Attendance'}

Internal\_1\_marks = Student\_Marks.objects.filter(Roll\_Number\_Marks = Roll\_Number).filter(Marks\_Type = Marks\_Rank[Exam]).filter(Semester = Semester\_Rank[Semester])

if len(Internal\_1\_marks) != 0:

content ={

'Student' : Internal\_1\_marks,

'Roll\_Number' : Roll\_Number,

'Group' : Group,

'Attendance' : Semesters,

'Marks' : Marks,

'sem' : Semester

}

return render(request,'temp/Studentfifth.html',content)

else:

Text = 'No Marks Posted Yet'

content ={

'Student' : Internal\_1\_marks,

'Roll\_Number' : Roll\_Number,

'Group' : Group,

'Attendance' : Semesters,

'Marks' : Marks,

'sem' : Semester,

'text' : Text

}

return render(request,'temp/Studentsixth.html',content)

ADD STUDENT.HTML

{% extends "main.html" %}

{% block content %}

{% load static%}

<style>

</style>

<a href= "/"><img src="{% static '/Image/Logouticon.jfif' %}" style="width:40px;height:40px; margin-left: 100%"></a><br>

<a href="javascript:history.go(-1)"><img src="{% static '/Image/Backicon.png' %}" style="width:40px;height:40px;margin-top: -70px"></a><br>

<img src ="{% static '/Image/student.jpeg' %}" style="width:150px; margin-left: 1200px; margin-top:10px;"><br>

<h1 style="font-family: serif; margin-top: -20px">ADD STUDENT</h1><br>

<form action = "/Faculty/AddStudent/Details/Register/NewStudent/" enctype='multipart/form-data' method = 'POST'>

<center>

<table style="margin-top:-100px; border:2px; width:90%;box-shadow: 0 3px 10px rgb(0 0 0 / 0.2)"><br><br>

{% csrf\_token %}

<br>

{{form}}

<br>

</table><br>

<input type="submit" value="Submit" class="btn btn-dark">

</center>

<form>

<br>

{% endblock %}

CERTIFICATES

{% extends "main.html" %}

{% block content %}

{% load static %}

<br>

<div class="container">

<a href= "/"><img src="{% static '/Image/Logouticon.jfif' %}" style="width:40px;height:40px; margin-left: 100%"></a><br>

<a href="javascript:history.go(-1)"><img src="{% static '/Image/Backicon.png' %}" style="width:40px;height:40px;margin-top: -70px"></a><br>

{% for student in Details %}

<h3>STUDY OF CONDUCT</h3>

<center><img src="{{student.Study\_Of\_Conduct.url}}" height="1000"></center>

<b><hr></b>

<br>

<h3>TRANSFER CERTIFICATE</h3>

<center><img src="{{student.Transfer\_Certificate.url}}" height="1000"></center>

<b><hr></b>

<br>

<h3>TENTH MARKS MEMO</h3>

<center><img src="{{student.Study\_Of\_Conduct.url}}" height="1000"></center>

<b><hr></b>

<br>

<h3>INTERMEDIATE MARKS MEMO</h3>

<center><img src="{{student.Study\_Of\_Conduct.url}}" height="1000"></center>

{% endfor %}

</div>

{% endblock %}

CLASS.HTML

{% extends "main.html" %}

{% block content %}

{%load static %}

<br>

<a href= "/"><img src="{% static '/Image/Logouticon.jfif' %}" style="width:40px;height:40px; margin-left: 100%"></a><br>

<a href="javascript:history.go(-1)"><img src="{% static '/Image/Backicon.png' %}" style="width:40px;height:40px;margin-top: -70px"></a><br>

<div class="container">

<center><h1 style="font-family: Candara;"> STUDENTS LIST</h1></center>

<div class="row row-cols-3">

{% for Student in Student\_Filter %}

<div class="col"><br>

<div class="card" style="width: 20rem; height: 23rem; box-shadow: 0 3px 10px rgb(0 0 0 / 0.2);"><br>

<center><img src="{{Student.Student\_Image.url}}" class="card-img-top" alt="..." style="width:200px;">

<div class="card-body">

<a href="/Faculty/{{Student.Year\_Of\_Admission}}/{{Student.First\_Name}}/{{Student.Roll\_Number}}/"><button class='btn btn-warning'>{{Student.Roll\_Number}}</button></a>

<p class="card-text"><b>{{Student.First\_Name}}&nbsp;{{Student.Last\_Name}}</b></p>

</div>

</div>

</div>

{% endfor %}

</div>

</div>

</div>

{% endblock %}

COURSES.HTML

{% extends "main.html" %}

{% block content %}

{% load static%}

<style>

.card{

box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);

border-color: #cb7b0b;

}

.a{

color: yellow;

}

</style>

<br>

<a href= "/"><img src="{% static '/Image/Logouticon.jfif' %}" style="width:40px;height:40px; margin-left: 100%"></a><br>

<a href="javascript:history.go(-1)"><img src="{% static '/Image/Backicon.png' %}" style="width:40px;height:40px;margin-top: -70px"></a><br>

<img src="{% static '\Image\college.jpeg' %}" style="width:10%;">

<a href="/Faculty/AddStudent/Details/Register/NewStudent/" style="margin-left:1000px;" ><button class="btn btn-warning">Add Student</button></a><br>

<div class="container1">

<br>

{% for Group in Groups\_list %}

<div class="row">

<div class='card' style="padding:20px solid #f7bd6d;">

{% for Year in Year\_Of\_Admission\_list %}

<div class="col">

<b><a href="/Faculty/{{Year}}/{{Group}}" style="color:#cb7b0b;">{{Group}}</a><br></b>

{{Year}}<br>

</div>

{% endfor %}

</div>

</div>

<br>

{% endfor %}

</div>

{% endblock %}