2110030107

Mallina Manjunadha

Introduction to Django Architecture

The Django is a free and open-source web application framework that is written in Python language. This framework is used in place of servlets, PHP, javascript to build web application backend part. The initial version of Django is released on 15th July 2005, which is developed by Django Software Foundation. The recent release of version 2.2.7 of the Django framework was done on 4th November 2019. Now we will learn about Django architecture with MVT

The main advantages of Django is to make the creation of complicated database included web applications as easy as possible, it is fast, many components are available implicitly, scalability and good security. Now, getting into the architecture of Django; it follows MVT.

Django Architecture

As mentioned, Django follows the MVT framework for architecture.

- M stands for Model
- V stands for View

- T stands for Template

MVT is generally very similar to that of MVC which is a Model, View, and Controller. The difference between MVC and MVT here is the Django itself does the work done by the controller part in the MVC architecture. Django does this work of controller by using templates. Precisely, the template file is a mixture of HTML part and Django Template Language also known as DTL

The Template handles the UI and architecture part of an application. The view does the logical part of the application and interacts with the Model to get the data and in turn modifies the template accordingly. Here as already mentioned, Django works as a controller and gets a URL that is linked to the view part of the application and thus transports the user responses to the application. This complete interaction is dealt with this Django MVT architecture. When we create a project, there would some default files that would be created.

I used the above command to create a new project in my my_projects folder. Now let me show you the files that got created in the empty folder

once the above command has been executed. These are the files that got created under the project first that we created.

There is, in turn, one more folder and a manage.py file that has been created. Now going into the first folder, we can observe the below files.

So the above files are those that got created automatically once a new project has been created.

- urls.py: As we know that our web page has to deal with many links, all the mappings from one page to others can be done here.
- wsgi.py: This is used to deploy our project.
- manage.py: Gives us a URL where the project can be displayed.

Django Architecture Model

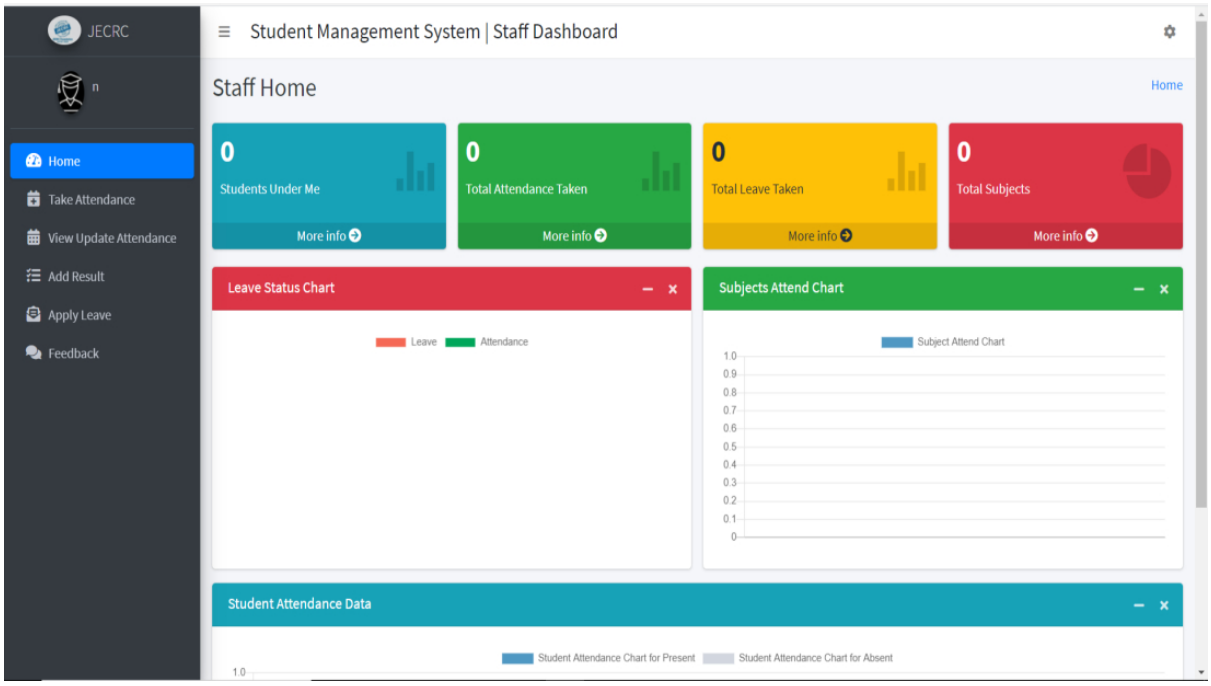
- In Django, the model does the linking to the database and each model gets mapped to a single table in the database. These fields and methods are declared under the file models.py
- With this linking to the database, we can actually each and every record or row from that particular table and can perform the DML operations on the table.
- Django.db.models.The model is the subclass that is used here. We can use the import statement by defining as from django.db import models.
- So after defining our database tables, columns and records; we are going to get the data linked to our application by defining the mapping in settings.py file under the INSTALLED_APPS.

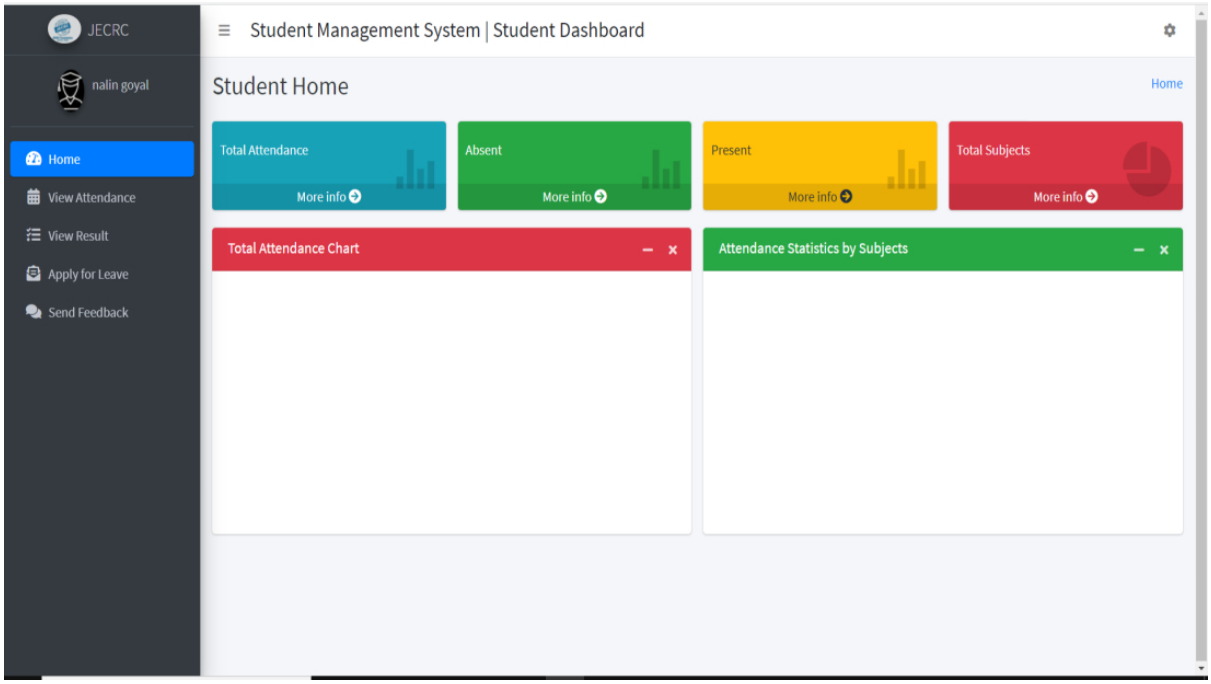
Django View

- This is the part where actually we would be mentioning our logic. This coding is done through the python file views.py
- This view also sends responses to the user when the application is used, to understand briefly, we can say that this view.py can deal with HttpResponse.

- Now, after creating a view, how can we link it to our application? How do you think that the system is going to understand to display a particular view? This can be done by mapping the views.py in urls.py file. As already mentioned, urls.py keeps track of all those different pages that we created and hence map each of them.

2ans






```
from django.shortcuts import render,HttpResponse, redirect,HttpResponseRedirect
from django.contrib.auth import logout, authenticate, login
from .models import CustomUser, Staffs, Students, AdminHOD
from django.contrib import messages

def home(request):
    return render(request, 'home.html')


def contact(request):
    return render(request, 'contact.html')


def loginUser(request):
    return render(request, 'login_page.html')

def doLogin(request):

    print("here")
    email_id = request.GET.get('email')
    password = request.GET.get('password')
    # user_type = request.GET.get('user_type')
    print(email_id)
    print(password)
    print(request.user)
```

```
    if not (email_id and password):
        messages.error(request, "Please provide all the details!!")
        return render(request, 'login_page.html')

    user = CustomUser.objects.filter(email=email_id, password=password).last()
    if not user:
        messages.error(request, 'Invalid Login Credentials!!')
        return render(request, 'login_page.html')

    login(request, user)
    print(request.user)

    if user.user_type == CustomUser.STUDENT:
        return redirect('student_home/')
    elif user.user_type == CustomUser.STAFF:
        return redirect('staff_home/')
    elif user.user_type == CustomUser.HOD:
        return redirect('admin_home/')

    return render(request, 'home.html')


def registration(request):
    return render(request, 'registration.html')


def doRegistration(request):
    first_name = request.GET.get('first_name')
    last_name = request.GET.get('last_name')
    email_id = request.GET.get('email')
    password = request.GET.get('password')
    confirm_password = request.GET.get('confirmPassword')

    print(email_id)
    print(password)
    print(confirm_password)
    print(first_name)
    print(last_name)
    if not (email_id and password and confirm_password):
        messages.error(request, 'Please provide all the details!!')
        return render(request, 'registration.html')

    if password != confirm_password:
        messages.error(request, 'Both passwords should match!!')
        return render(request, 'registration.html')
```

```
    is_user_exists = CustomUser.objects.filter(email=email_id).exists()

    if is_user_exists:
        messages.error(request, 'User with this email id already exists. Please
proceed to login!!')
        return render(request, 'registration.html')

    user_type = get_user_type_from_email(email_id)

    if user_type is None:
        messages.error(request, "Please use valid format for the email id:
'<username>.<staff|student|hod>@<college_domain>'")
        return render(request, 'registration.html')

    username = email_id.split('@')[0].split('.')[0]

    if CustomUser.objects.filter(username=username).exists():
        messages.error(request, 'User with this username already exists. Please use
different username')
        return render(request, 'registration.html')

    user = CustomUser()
    user.username = username
    user.email = email_id
    user.password = password
    user.user_type = user_type
    user.first_name = first_name
    user.last_name = last_name
    user.save()

    if user_type == CustomUser.STAFF:
        Staffs.objects.create(admin=user)
    elif user_type == CustomUser.STUDENT:
        Students.objects.create(admin=user)
    elif user_type == CustomUser.HOD:
        AdminHOD.objects.create(admin=user)
    return render(request, 'login_page.html')


def logout_user(request):
    logout(request)
    return HttpResponseRedirect('/')


def get_user_type_from_email(email_id):
    """
```

```
    Returns CustomUser.user_type corresponding to the given email address
    email_id should be in following format:
    '<username>.<staff|student|hod>@<college_domain>'
    eg.: 'abhishek.staff@jecrc.com'
    """

    try:
        email_id = email_id.split('@')[0]
        email_user_type = email_id.split('.')[1]
        return CustomUser.EMAIL_TO_USER_TYPE_MAP[email_user_type]
    except:
        return None
```

Step 8: Create or Go to **urls.py** of student_management_app and add the following URLs.

- Python3

```
from django.contrib import admin
from django.urls import path, include
from . import views
from .import HodViews, StaffViews, StudentViews

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home, name="home"),
    path('contact', views.contact, name="contact"),
    path('login', views.loginUser, name="login"),
    path('logout_user', views.logout_user, name="logout_user"),
    path('registration', views.registration, name="registration"),
    path('doLogin', views.doLogin, name="doLogin"),
    path('doRegistration', views.doRegistration, name="doRegistration"),

      # URLS for Student
    path('student_home/', StudentViews.student_home, name="student_home"),
    path('student_view_attendance/', StudentViews.student_view_attendance,
name="student_view_attendance"),
    path('student_view_attendance_post/',
StudentViews.student_view_attendance_post, name="student_view_attendance_post"),
    path('student_apply_leave/', StudentViews.student_apply_leave,
name="student_apply_leave"),
    path('student_apply_leave_save/', StudentViews.student_apply_leave_save,
name="student_apply_leave_save"),
    path('student_feedback/', StudentViews.student_feedback,
name="student_feedback"),
    path('student_feedback_save/', StudentViews.student_feedback_save,
name="student_feedback_save"),
```

```
    path('student_profile/', StudentViews.student_profile, name="student_profile"),
    path('student_profile_update/', StudentViews.student_profile_update,
name="student_profile_update"),
    path('student_view_result/', StudentViews.student_view_result,
name="student_view_result"),


     # URLS for Staff
    path('staff_home/', StaffViews.staff_home, name="staff_home"),
    path('staff_take_attendance/', StaffViews.staff_take_attendance,
name="staff_take_attendance"),
    path('get_students/', StaffViews.get_students, name="get_students"),
    path('save_attendance_data/', StaffViews.save_attendance_data,
name="save_attendance_data"),
    path('staff_update_attendance/', StaffViews.staff_update_attendance,
name="staff_update_attendance"),
    path('get_attendance_dates/', StaffViews.get_attendance_dates,
name="get_attendance_dates"),
    path('get_attendance_student/', StaffViews.get_attendance_student,
name="get_attendance_student"),
    path('update_attendance_data/', StaffViews.update_attendance_data,
name="update_attendance_data"),
    path('staff_apply_leave/', StaffViews.staff_apply_leave,
name="staff_apply_leave"),
    path('staff_apply_leave_save/', StaffViews.staff_apply_leave_save,
name="staff_apply_leave_save"),
    path('staff_feedback/', StaffViews.staff_feedback, name="staff_feedback"),
    path('staff_feedback_save/', StaffViews.staff_feedback_save,
name="staff_feedback_save"),
    path('staff_profile/', StaffViews.staff_profile, name="staff_profile"),
    path('staff_profile_update/', StaffViews.staff_profile_update,
name="staff_profile_update"),
    path('staff_add_result/', StaffViews.staff_add_result,
name="staff_add_result"),
    path('staff_add_result_save/', StaffViews.staff_add_result_save,
name="staff_add_result_save"),

    # URL for Admin
    path('admin_home/', HodViews.admin_home, name="admin_home"),
    path('add_staff/', HodViews.add_staff, name="add_staff"),
    path('add_staff_save/', HodViews.add_staff_save, name="add_staff_save"),
    path('manage_staff/', HodViews.manage_staff, name="manage_staff"),
    path('edit_staff/<staff_id>/', HodViews.edit_staff, name="edit_staff"),
    path('edit_staff_save/', HodViews.edit_staff_save, name="edit_staff_save"),
    path('delete_staff/<staff_id>/', HodViews.delete_staff, name="delete_staff"),
    path('add_course/', HodViews.add_course, name="add_course"),
```

```
    path('add_course_save/', HodViews.add_course_save, name="add_course_save"),
    path('manage_course/', HodViews.manage_course, name="manage_course"),
    path('edit_course/<course_id>/', HodViews.edit_course, name="edit_course"),
    path('edit_course_save/', HodViews.edit_course_save, name="edit_course_save"),
    path('delete_course/<course_id>/', HodViews.delete_course,
name="delete_course"),
    path('manage_session/', HodViews.manage_session, name="manage_session"),
    path('add_session/', HodViews.add_session, name="add_session"),
    path('add_session_save/', HodViews.add_session_save, name="add_session_save"),
    path('edit_session/<session_id>', HodViews.edit_session, name="edit_session"),
    path('edit_session_save/', HodViews.edit_session_save,
name="edit_session_save"),
    path('delete_session/<session_id>/', HodViews.delete_session,
name="delete_session"),
    path('add_student/', HodViews.add_student, name="add_student"),
    path('add_student_save/', HodViews.add_student_save, name="add_student_save"),
    path('edit_student/<student_id>', HodViews.edit_student, name="edit_student"),
    path('edit_student_save/', HodViews.edit_student_save,
name="edit_student_save"),
    path('manage_student/', HodViews.manage_student, name="manage_student"),
    path('delete_student/<student_id>/', HodViews.delete_student,
name="delete_student"),
    path('add_subject/', HodViews.add_subject, name="add_subject"),
    path('add_subject_save/', HodViews.add_subject_save, name="add_subject_save"),
    path('manage_subject/', HodViews.manage_subject, name="manage_subject"),
    path('edit_subject/<subject_id>/', HodViews.edit_subject, name="edit_subject"),
    path('edit_subject_save/', HodViews.edit_subject_save,
name="edit_subject_save"),
    path('delete_subject/<subject_id>/', HodViews.delete_subject,
name="delete_subject"),
    path('check_email_exist/', HodViews.check_email_exist,
name="check_email_exist"),
    path('check_username_exist/', HodViews.check_username_exist,
name="check_username_exist"),
    path('student_feedback_message/', HodViews.student_feedback_message,
name="student_feedback_message"),
    path('student_feedback_message_reply/',
HodViews.student_feedback_message_reply, name="student_feedback_message_reply"),
    path('staff_feedback_message/', HodViews.staff_feedback_message,
name="staff_feedback_message"),
    path('staff_feedback_message_reply/', HodViews.staff_feedback_message_reply,
name="staff_feedback_message_reply"),
    path('student_leave_view/', HodViews.student_leave_view,
name="student_leave_view"),
    path('student_leave_approve/<leave_id>/', HodViews.student_leave_approve,
name="student_leave_approve"),
```

```
    path('student_leave_reject/<leave_id>/', HodViews.student_leave_reject,
name="student_leave_reject"),
    path('staff_leave_view/', HodViews.staff_leave_view, name="staff_leave_view"),
    path('staff_leave_approve/<leave_id>/', HodViews.staff_leave_approve,
name="staff_leave_approve"),
    path('staff_leave_reject/<leave_id>/', HodViews.staff_leave_reject,
name="staff_leave_reject"),
    path('admin_view_attendance/', HodViews.admin_view_attendance,
name="admin_view_attendance"),
    path('admin_get_attendance_dates/', HodViews.admin_get_attendance_dates,
name="admin_get_attendance_dates"),
    path('admin_get_attendance_student/', HodViews.admin_get_attendance_student,
name="admin_get_attendance_student"),
    path('admin_profile/', HodViews.admin_profile, name="admin_profile"),
    path('admin_profile_update/', HodViews.admin_profile_update,
name="admin_profile_update"),

]
```

Step 9: Now create a file **StudentViews.py**. It contains the views that are used on the student Interface.

- Python3

```
from django.shortcuts import render, redirect
from django.http import HttpResponse, HttpResponseRedirect
from django.contrib import messages
from django.core.files.storage import FileSystemStorage
from django.urls import reverse
import datetime
from .models import CustomUser, Staffs, Courses, Subjects, Students, Attendance,
AttendanceReport, LeaveReportStudent, FeedBackStudent, StudentResult

def student_home(request):
  student_obj = Students.objects.get(admin=request.user.id)
  total_attendance =    AttendanceReport.objects.filter(student_id=student_obj).count(
  attendance_present = AttendanceReport.objects.filter(student_id=student_obj,
                                                       status=True).count()
  attendance_absent =  AttendanceReport.objects.filter(student_id=student_obj,
                                                       status=False).count()
  course_obj = Courses.objects.get(id=student_obj.course_id.id)
  total_subjects = Subjects.objects.filter(course_id=course_obj).count()
  subject_name = []
  data_present = []
  data_absent = []
  subject_data = Subjects.objects.filter(course_id=student_obj.course_id)
  for subject in subject_data:
```

```
        attendance = Attendance.objects.filter(subject_id=subject.id)
        attendance_present_count = AttendanceReport.objects.filter(attendance_id__in=atte
                                                                   status=True,
                                                                   student_id=student_obj
        attendance_absent_count = AttendanceReport.objects.filter(attendance_id__in=atten
                                                                  status=False,
                                                                  student_id=student_obj.
        subject_name.append(subject.subject_name)
        data_present.append(attendance_present_count)
        data_absent.append(attendance_absent_count)

        context={
            "total_attendance": total_attendance,
            "attendance_present": attendance_present,
            "attendance_absent": attendance_absent,
            "total_subjects": total_subjects,
            "subject_name": subject_name,
            "data_present": data_present,
            "data_absent": data_absent
        }
        return render(request, "student_template/student_home_template.html")


    def student_view_attendance(request):

        # Getting Logged in Student Data
        student = Students.objects.get(admin=request.user.id)

        # Getting Course Enrolled of LoggedIn Student
        course = student.course_id

        # Getting the Subjects of Course Enrolled
        subjects = Subjects.objects.filter(course_id=course)
        context = {
            "subjects": subjects
        }
        return render(request, "student_template/student_view_attendance.html", context)


    def student_view_attendance_post(request):
        if request.method != "POST":
            messages.error(request, "Invalid Method")
            return redirect('student_view_attendance')
        else:
            # Getting all the Input Data
            subject_id = request.POST.get('subject')
```

```
        start_date = request.POST.get('start_date')
        end_date = request.POST.get('end_date')

        # Parsing the date data into Python object
        start_date_parse = datetime.datetime.strptime(start_date, '%Y-%m-%d').date()
        end_date_parse = datetime.datetime.strptime(end_date, '%Y-%m-%d').date()

        # Getting all the Subject Data based on Selected Subject
        subject_obj = Subjects.objects.get(id=subject_id)

        # Getting Logged In User Data
        user_obj = CustomUser.objects.get(id=request.user.id)

        # Getting Student Data Based on Logged in Data
        stud_obj = Students.objects.get(admin=user_obj)

        # Now Accessing Attendance Data based on the Range of Date
        # Selected and Subject Selected
        attendance = Attendance.objects.filter(attendance_date__range=(start_date_par
                                                                       end_date_parse
                                               subject_id=subject_obj)
        # Getting Attendance Report based on the attendance
        # details obtained above
        attendance_reports = AttendanceReport.objects.filter(attendance_id__in=attend
                                                             student_id=stud_obj)


        context = {
            "subject_obj": subject_obj,
            "attendance_reports": attendance_reports
        }

        return render(request, 'student_template/student_attendance_data.html', conte


def student_apply_leave(request):
    student_obj = Students.objects.get(admin=request.user.id)
    leave_data = LeaveReportStudent.objects.filter(student_id=student_obj)
    context = {
        "leave_data": leave_data
    }
    return render(request, 'student_template/student_apply_leave.html', context)


def student_apply_leave_save(request):
    if request.method != "POST":
```

```
            messages.error(request, "Invalid Method")
            return redirect('student_apply_leave')
        else:
            leave_date = request.POST.get('leave_date')
            leave_message = request.POST.get('leave_message')

            student_obj = Students.objects.get(admin=request.user.id)
            try:
                leave_report = LeaveReportStudent(student_id=student_obj,
                                                  leave_date=leave_date,
                                                  leave_message=leave_message,
                                                  leave_status=0)
                leave_report.save()
                messages.success(request, "Applied for Leave.")
                return redirect('student_apply_leave')
            except:
                messages.error(request, "Failed to Apply Leave")
                return redirect('student_apply_leave')


def student_feedback(request):
    student_obj = Students.objects.get(admin=request.user.id)
    feedback_data = FeedBackStudent.objects.filter(student_id=student_obj)
    context = {
        "feedback_data": feedback_data
    }
    return render(request, 'student_template/student_feedback.html', context)


def student_feedback_save(request):
    if request.method != "POST":
        messages.error(request, "Invalid Method.")
        return redirect('student_feedback')
    else:
        feedback = request.POST.get('feedback_message')
        student_obj = Students.objects.get(admin=request.user.id)

        try:
            add_feedback = FeedBackStudent(student_id=student_obj,
                                           feedback=feedback,
                                           feedback_reply="")
            add_feedback.save()
            messages.success(request, "Feedback Sent.")
            return redirect('student_feedback')
        except:
            messages.error(request, "Failed to Send Feedback.")
```

```
            return redirect('student_feedback')


def student_profile(request):
    user = CustomUser.objects.get(id=request.user.id)
    student = Students.objects.get(admin=user)

    context={
        "user": user,
        "student": student
    }
    return render(request, 'student_template/student_profile.html', context)


def student_profile_update(request):
    if request.method != "POST":
        messages.error(request, "Invalid Method!")
        return redirect('student_profile')
    else:
        first_name = request.POST.get('first_name')
        last_name = request.POST.get('last_name')
        password = request.POST.get('password')
        address = request.POST.get('address')

        try:
            customuser = CustomUser.objects.get(id=request.user.id)
            customuser.first_name = first_name
            customuser.last_name = last_name
            if password != None and password != "":
                customuser.set_password(password)
            customuser.save()

            student = Students.objects.get(admin=customuser.id)
            student.address = address
            student.save()

            messages.success(request, "Profile Updated Successfully")
            return redirect('student_profile')
        except:
            messages.error(request, "Failed to Update Profile")
            return redirect('student_profile')


def student_view_result(request):
    student = Students.objects.get(admin=request.user.id)
    student_result = StudentResult.objects.filter(student_id=student.id)
```

```
    context = {
        "student_result": student_result,
    }
    return render(request, "student_template/student_view_result.html", context)
```

Step 10: Now add the **StaffViews.py**. It contains the views of the staff interface.