College Management System using Django – Python Project

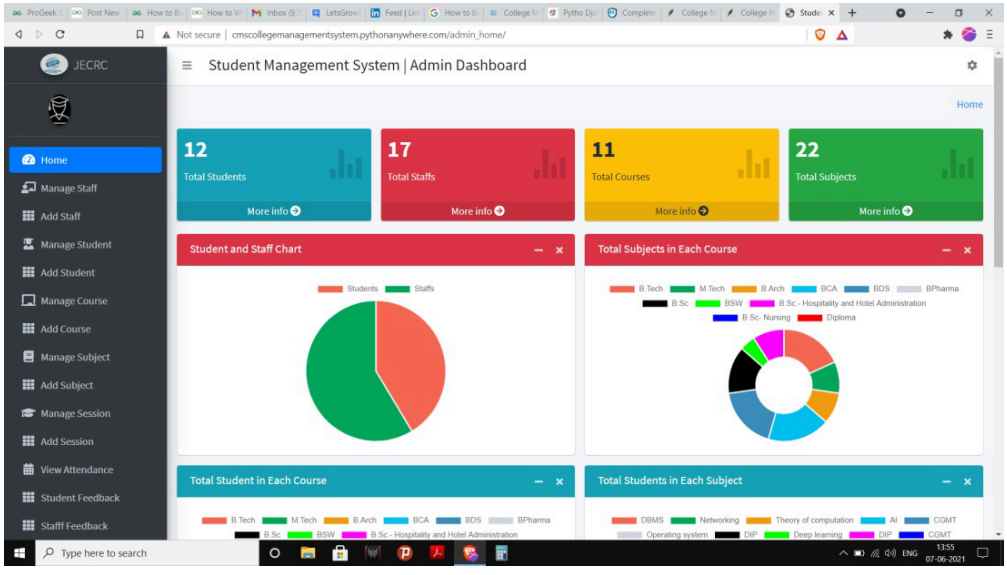
Last Updated : 10 Sep, 2024

In this article, we are going to build College Management System using *Django* and will be using *dbsqlite* database. In the times of covid, when education has totally become digital, there comes a need for a system that can connect teachers, students, and HOD and that was the motivation behind building this project.

This project allows HOD, staff, and students to register themselves. This project has three interfaces.

1. For Head Of Department(HOD):

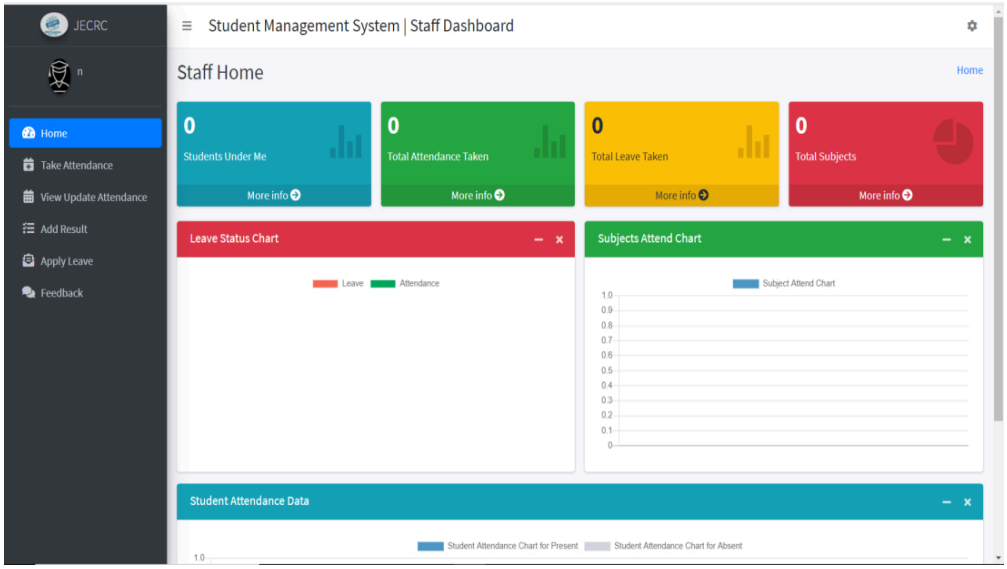
The below image shows the interface for the Head of Departments of the College:



HOD Interface

2. For Staff:

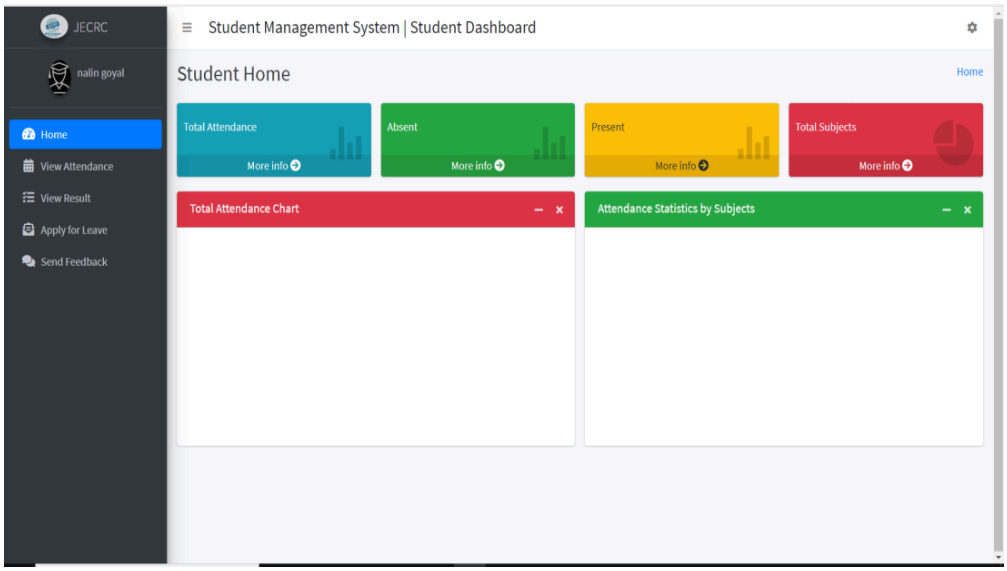
The below image shows the interface for the Staff of the College:



Staff Interface

3. For Students:

The below image shows the interface for the Students of the College:



Student Interface

Tools and Technologies Used in the Project:

1. HTML
2. CSS
3. JAVASCRIPT
4. JQUERY

5. BOOTSTRAP
6. DJANGO

Required Skillset to Build the Project:

- Basic knowledge of HTML, CSS, JavaScript, and Django.

Step By Step Implementation

Follow the below steps to implement the discussed project:

Step 1: Install Django.

Step 2: Create a folder with the name ***College_Management_System*** and open it with VS Code.

Step 3: Open the terminal and create a new project "student_management_project" using the below command.

```
django-admin startproject student_management_project
```

Step 4: Enter inside the folder "student_management_project" and create the app "student_management_app".

```
python manage.py startapp student_management_app
```

Step 5: Go to student_management_project *-> settings.py -> INSTALLED_APPS* and add our app **'student_management_app'**.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'student_management_app',
]
```

Step 6: Go to urls.py of student_management_project and add the below path in ***urlpatterns***. (Note – Import include as **from django.urls import path, include**)

```
path('', include('student_management_app.urls'))
```

Step 7: Now enter the views that are going to use in **views.py** of student_management_app.

Python

```python
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

    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.

Python

```
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"),
```

```
16        # URLS for Student
17      path('student_home/', StudentViews.student_home,
    name="student_home"),
18      path('student_view_attendance/',
    StudentViews.student_view_attendance,
    name="student_view_attendance"),
19      path('student_view_attendance_post/',
    StudentViews.student_view_attendance_post,
    name="student_view_attendance_post"),
20      path('student_apply_leave/',
    StudentViews.student_apply_leave,
    name="student_apply_leave"),
21      path('student_apply_leave_save/',
    StudentViews.student_apply_leave_save,
    name="student_apply_leave_save"),
22      path('student_feedback/',
    StudentViews.student_feedback, name="student_feedback"),
23      path('student_feedback_save/',
    StudentViews.student_feedback_save,
    name="student_feedback_save"),
24      path('student_profile/', StudentViews.student_profile,
    name="student_profile"),
25      path('student_profile_update/',
    StudentViews.student_profile_update,
    name="student_profile_update"),
26      path('student_view_result/',
    StudentViews.student_view_result,
    name="student_view_result"),
27
28
29       # URLS for Staff
30      path('staff_home/', StaffViews.staff_home,
    name="staff_home"),
31      path('staff_take_attendance/',
    StaffViews.staff_take_attendance,
    name="staff_take_attendance"),
32      path('get_students/', StaffViews.get_students,
    name="get_students"),
33      path('save_attendance_data/',
    StaffViews.save_attendance_data,
    name="save_attendance_data"),
34      path('staff_update_attendance/',
    StaffViews.staff_update_attendance,
    name="staff_update_attendance"),
35      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"),
82      path('student_feedback_message_reply/',
HodViews.student_feedback_message_reply,
name="student_feedback_message_reply"),
83      path('staff_feedback_message/',
HodViews.staff_feedback_message,
name="staff_feedback_message"),
84      path('staff_feedback_message_reply/',
HodViews.staff_feedback_message_reply,
name="staff_feedback_message_reply"),
85      path('student_leave_view/',
HodViews.student_leave_view, name="student_leave_view"),
86      path('student_leave_approve/<leave_id>/',
HodViews.student_leave_approve,
name="student_leave_approve"),
87      path('student_leave_reject/<leave_id>/',
HodViews.student_leave_reject,
name="student_leave_reject"),
88      path('staff_leave_view/', HodViews.staff_leave_view,
name="staff_leave_view"),
89      path('staff_leave_approve/<leave_id>/',
HodViews.staff_leave_approve, name="staff_leave_approve"),
90      path('staff_leave_reject/<leave_id>/',
HodViews.staff_leave_reject, name="staff_leave_reject"),
91      path('admin_view_attendance/',
HodViews.admin_view_attendance,
name="admin_view_attendance"),
92      path('admin_get_attendance_dates/',
HodViews.admin_get_attendance_dates,
name="admin_get_attendance_dates"),
93      path('admin_get_attendance_student/',
HodViews.admin_get_attendance_student,
name="admin_get_attendance_student"),
94      path('admin_profile/', HodViews.admin_profile,
name="admin_profile"),
95      path('admin_profile_update/',
HodViews.admin_profile_update,
name="admin_profile_update"),
96
97  ]
```

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

Python

```python
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=attendance, status=True, student_id=student_obj.id).count()
```

```
    attendance_absent_count =
AttendanceReport.objects.filter(attendance_id__in=attendan
ce,
28
status=False,
29
student_id=student_obj.id).count()
30      subject_name.append(subject.subject_name)
31      data_present.append(attendance_present_count)
32      data_absent.append(attendance_absent_count)
33
34      context={
35          "total_attendance": total_attendance,
36          "attendance_present": attendance_present,
37          "attendance_absent": attendance_absent,
38          "total_subjects": total_subjects,
39          "subject_name": subject_name,
40          "data_present": data_present,
41          "data_absent": data_absent
42      }
43      return render(request,
"student_template/student_home_template.html")
44
45
46  def student_view_attendance(request):
47
48      # Getting Logged in Student Data
49      student = Students.objects.get(admin=request.user.id)
50
51      # Getting Course Enrolled of LoggedIn Student
52      course = student.course_id
53
54      # Getting the Subjects of Course Enrolled
55      subjects = Subjects.objects.filter(course_id=course)
56      context = {
57          "subjects": subjects
58      }
59      return render(request,
"student_template/student_view_attendance.html", context)
60
61
62  def student_view_attendance_post(request):
63      if request.method != "POST":
64          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_parse,

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=attendance,

student_id=stud_obj)


        context = {
```

```
            "subject_obj": subject_obj,
            "attendance_reports": attendance_reports
        }

        return render(request, 'student_template/student_attendance_data.html', context)


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.

Python

```python
from django.shortcuts import render, redirect
from django.http import HttpResponse, HttpResponseRedirect, JsonResponse
from django.contrib import messages
from django.core.files.storage import FileSystemStorage
from django.urls import reverse
from django.views.decorators.csrf import csrf_exempt
from django.core import serializers
import json


from .models import CustomUser, Staffs, Courses, Subjects, Students, SessionYearModel, Attendance, AttendanceReport, LeaveReportStaff, FeedBackStaffs, StudentResult


def staff_home(request):

    # Fetching All Students under Staff
    print(request.user.id)
    subjects = Subjects.objects.filter(staff_id=request.user.id)
    print(subjects)
    course_id_list = []
    for subject in subjects:
        course = Courses.objects.get(id=subject.course_id.id)
        course_id_list.append(course.id)

    final_course = []
```

```
        # Removing Duplicate Course Id
27      for course_id in course_id_list:
28          if course_id not in final_course:
29              final_course.append(course_id)
30
31      print(final_course)
32      students_count =
    Students.objects.filter(course_id__in=final_course).count(
    )
33      subject_count = subjects.count()
34      print(subject_count)
35      print(students_count)
36
37      # Fetch All Attendance Count
38      attendance_count =
    Attendance.objects.filter(subject_id__in=subjects).count()
39
40      # Fetch All Approve Leave
41      # print(request.user)
42      print(request.user.user_type)
43      staff = Staffs.objects.get(admin=request.user.id)
44      leave_count =
    LeaveReportStaff.objects.filter(staff_id=staff.id,
45
    leave_status=1).count()
46
47      # Fetch Attendance Data by Subjects
48      subject_list = []
49      attendance_list = []
50      for subject in subjects:
51          attendance_count1 =
    Attendance.objects.filter(subject_id=subject.id).count()
52          subject_list.append(subject.subject_name)
53          attendance_list.append(attendance_count1)
54
55      students_attendance =
    Students.objects.filter(course_id__in=final_course)
56      student_list = []
57      student_list_attendance_present = []
58      student_list_attendance_absent = []
59      for student in students_attendance:
60          attendance_present_count =
    AttendanceReport.objects.filter(status=True,
61
    student_id=student.id).count()
```

```
        attendance_absent_count =
AttendanceReport.objects.filter(status=False,
63
student_id=student.id).count()
64          student_list.append(student.admin.first_name+" "+
student.admin.last_name)
65
student_list_attendance_present.append(attendance_present_
count)
66
student_list_attendance_absent.append(attendance_absent_co
unt)
67
68      context={
69          "students_count": students_count,
70          "attendance_count": attendance_count,
71          "leave_count": leave_count,
72          "subject_count": subject_count,
73          "subject_list": subject_list,
74          "attendance_list": attendance_list,
75          "student_list": student_list,
76          "attendance_present_list":
student_list_attendance_present,
77          "attendance_absent_list":
student_list_attendance_absent
78      }
79      return render(request,
"staff_template/staff_home_template.html", context)
80
81
82
83  def staff_take_attendance(request):
84      subjects =
Subjects.objects.filter(staff_id=request.user.id)
85      session_years = SessionYearModel.objects.all()
86      context = {
87          "subjects": subjects,
88          "session_years": session_years
89      }
90      return render(request,
"staff_template/take_attendance_template.html", context)
91
92
93  def staff_apply_leave(request):
94      print(request.user.id)
```

```
    staff_obj = Staffs.objects.get(admin=request.user.id)
    leave_data = LeaveReportStaff.objects.filter(staff_id=staff_obj)
    context = {
        "leave_data": leave_data
    }
    return render(request, "staff_template/staff_apply_leave_template.html", context)


def staff_apply_leave_save(request):
    if request.method != "POST":
        messages.error(request, "Invalid Method")
        return redirect('staff_apply_leave')
    else:
        leave_date = request.POST.get('leave_date')
        leave_message = request.POST.get('leave_message')

        staff_obj = Staffs.objects.get(admin=request.user.id)
        try:
            leave_report = LeaveReportStaff(staff_id=staff_obj,
leave_date=leave_date,
leave_message=leave_message,
leave_status=0)
            leave_report.save()
            messages.success(request, "Applied for Leave.")
            return redirect('staff_apply_leave')
        except:
            messages.error(request, "Failed to Apply Leave")
            return redirect('staff_apply_leave')


def staff_feedback(request):
  return render(request, "staff_template/staff_feedback_template.html")


def staff_feedback_save(request):
```

```
    if request.method != "POST":
        messages.error(request, "Invalid Method.")
        return redirect('staff_feedback')
    else:
        feedback = request.POST.get('feedback_message')
        staff_obj = Staffs.objects.get(admin=request.user.id)

        try:
            add_feedback = FeedBackStaffs(staff_id=staff_obj,
feedback=feedback,
feedback_reply="")
            add_feedback.save()
            messages.success(request, "Feedback Sent.")
            return redirect('staff_feedback')
        except:
            messages.error(request, "Failed to Send Feedback.")
            return redirect('staff_feedback')



@csrf_exempt
def get_students(request):

    subject_id = request.POST.get("subject")
    session_year = request.POST.get("session_year")

    # Students enroll to Course, Course has Subjects
    # Getting all data from subject model based on subject_id
    subject_model = Subjects.objects.get(id=subject_id)

    session_model = SessionYearModel.objects.get(id=session_year)

    students = Students.objects.filter(course_id=subject_model.course_id,
session_year_id=session_model)

    # Only Passing Student Id and Student Name Only
```

```
    list_data = []

    for student in students:
        data_small={"id":student.admin.id,
                    "name":student.admin.first_name+" "+student.admin.last_name}
        list_data.append(data_small)

    return JsonResponse(json.dumps(list_data), content_type="application/json", safe=False)




@csrf_exempt
def save_attendance_data(request):

    # Get Values from Staf Take Attendance form via AJAX (JavaScript)
    # Use getlist to access HTML Array/List Input Data
    student_ids = request.POST.get("student_ids")
    subject_id = request.POST.get("subject_id")
    attendance_date = request.POST.get("attendance_date")
    session_year_id = request.POST.get("session_year_id")

    subject_model = Subjects.objects.get(id=subject_id)
    session_year_model = SessionYearModel.objects.get(id=session_year_id)

    json_student = json.loads(student_ids)

    try:
        # First Attendance Data is Saved on Attendance Model
        attendance = Attendance(subject_id=subject_model,

attendance_date=attendance_date,

session_year_id=session_year_model)
        attendance.save()

        for stud in json_student:
            # Attendance of Individual Student saved on AttendanceReport Model
```

```
                student = Students.objects.get(admin=stud['id'])
203             attendance_report = AttendanceReport(student_id=student,
204 
    attendance_id=attendance,
205 
    status=stud['status'])
206             attendance_report.save()
207         return HttpResponse("OK")
208     except:
209         return HttpResponse("Error")
210 
211 
212 
213 
214 def staff_update_attendance(request):
215     subjects = Subjects.objects.filter(staff_id=request.user.id)
216     session_years = SessionYearModel.objects.all()
217     context = {
218         "subjects": subjects,
219         "session_years": session_years
220     }
221     return render(request, "staff_template/update_attendance_template.html", context)
222 
223 @csrf_exempt
224 def get_attendance_dates(request):
225 
226 
227     # Getting Values from Ajax POST 'Fetch Student'
228     subject_id = request.POST.get("subject")
229     session_year = request.POST.get("session_year_id")
230 
231     # Students enroll to Course, Course has Subjects
232     # Getting all data from subject model based on subject_id
233     subject_model = Subjects.objects.get(id=subject_id)
234 
235     session_model = SessionYearModel.objects.get(id=session_year)
236     attendance = Attendance.objects.filter(subject_id=subject_model,
```

```
    session_year_id=session_model)

    # Only Passing Student Id and Student Name Only
    list_data = []

    for attendance_single in attendance:
        data_small={"id":attendance_single.id,
"attendance_date":str(attendance_single.attendance_date),
"session_year_id":attendance_single.session_year_id.id}
        list_data.append(data_small)

    return JsonResponse(json.dumps(list_data),
                        content_type="application/json",
safe=False)


@csrf_exempt
def get_attendance_student(request):

    # Getting Values from Ajax POST 'Fetch Student'
    attendance_date = request.POST.get('attendance_date')
    attendance =
Attendance.objects.get(id=attendance_date)

    attendance_data =
AttendanceReport.objects.filter(attendance_id=attendance)
    # Only Passing Student Id and Student Name Only
    list_data = []

    for student in attendance_data:
        data_small={"id":student.student_id.admin.id,
"name":student.student_id.admin.first_name+"
"+student.student_id.admin.last_name,
"status":student.status}
        list_data.append(data_small)

    return JsonResponse(json.dumps(list_data),
                        content_type="application/json",
                        safe=False)


```

```
@csrf_exempt
def update_attendance_data(request):
    student_ids = request.POST.get("student_ids")

    attendance_date = request.POST.get("attendance_date")
    attendance = Attendance.objects.get(id=attendance_date)

    json_student = json.loads(student_ids)

    try:

        for stud in json_student:

            # Attendance of Individual Student saved on AttendanceReport Model
            student = Students.objects.get(admin=stud['id'])

            attendance_report = AttendanceReport.objects.get(student_id=student,

attendance_id=attendance)
            attendance_report.status=stud['status']

            attendance_report.save()
        return HttpResponse("OK")
    except:
        return HttpResponse("Error")


def staff_profile(request):
    user = CustomUser.objects.get(id=request.user.id)
    staff = Staffs.objects.get(admin=user)

    context={
        "user": user,
        "staff": staff
    }
    return render(request, 'staff_template/staff_profile.html', context)


def staff_profile_update(request):
```

```
    if request.method != "POST":
        messages.error(request, "Invalid Method!")
        return redirect('staff_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()

            staff = Staffs.objects.get(admin=customuser.id)
            staff.address = address
            staff.save()

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



def staff_add_result(request):
    subjects = Subjects.objects.filter(staff_id=request.user.id)
    session_years = SessionYearModel.objects.all()
    context = {
        "subjects": subjects,
        "session_years": session_years,
    }
    return render(request, "staff_template/add_result_template.html", context)

```

```
def staff_add_result_save(request):
    if request.method != "POST":
        messages.error(request, "Invalid Method")
        return redirect('staff_add_result')
    else:
        student_admin_id = request.POST.get('student_list')
        assignment_marks = request.POST.get('assignment_marks')
        exam_marks = request.POST.get('exam_marks')
        subject_id = request.POST.get('subject')

        student_obj = Students.objects.get(admin=student_admin_id)
        subject_obj = Subjects.objects.get(id=subject_id)

        try:
            # Check if Students Result Already Exists or not
            check_exist = StudentResult.objects.filter(subject_id=subject_obj,
student_id=student_obj).exists()
            if check_exist:
                result = StudentResult.objects.get(subject_id=subject_obj,
student_id=student_obj)
                result.subject_assignment_marks = assignment_marks
                result.subject_exam_marks = exam_marks
                result.save()
                messages.success(request, "Result Updated Successfully!")
                return redirect('staff_add_result')
            else:
                result = StudentResult(student_id=student_obj,
subject_id=subject_obj,
subject_exam_marks=exam_marks,
subject_assignment_marks=assignment_marks)
                result.save()
```

```
                messages.success(request, "Result Added
Successfully!")
382             return redirect('staff_add_result')
383         except:
384             messages.error(request, "Failed to Add
Result!")
385             return redirect('staff_add_result')
```

Step 11: Now add the **HodViews.py**. It contains the views of the HOD interface.

Python

```
1  from django.shortcuts import render, redirect
2  from django.http import HttpResponse,
   HttpResponseRedirect, JsonResponse
3  from django.contrib import messages
4  from django.core.files.storage import FileSystemStorage
5  from django.urls import reverse
6  from django.views.decorators.csrf import csrf_exempt
7  import json
8
9  from .forms import AddStudentForm, EditStudentForm
10
11 from .models import CustomUser, Staffs, Courses, Subjects,
   Students, SessionYearModel, FeedBackStudent,
   FeedBackStaffs, LeaveReportStudent, LeaveReportStaff,
   Attendance, AttendanceReport
12
13
14 def admin_home(request):
15
16     all_student_count = Students.objects.all().count()
17     subject_count = Subjects.objects.all().count()
18     course_count = Courses.objects.all().count()
19     staff_count = Staffs.objects.all().count()
20     course_all = Courses.objects.all()
21     course_name_list = []
22     subject_count_list = []
23     student_count_list_in_course = []
24
25     for course in course_all:
```

```
            subjects =
    Subjects.objects.filter(course_id=course.id).count()
27          students =
    Students.objects.filter(course_id=course.id).count()
28          course_name_list.append(course.course_name)
29          subject_count_list.append(subjects)
30          student_count_list_in_course.append(students)
31
32      subject_all = Subjects.objects.all()
33      subject_list = []
34      student_count_list_in_subject = []
35      for subject in subject_all:
36          course =
    Courses.objects.get(id=subject.course_id.id)
37          student_count =
    Students.objects.filter(course_id=course.id).count()
38          subject_list.append(subject.subject_name)
39
    student_count_list_in_subject.append(student_count)
40
41      # For Saffs
42      staff_attendance_present_list=[]
43      staff_attendance_leave_list=[]
44      staff_name_list=[]
45
46      staffs = Staffs.objects.all()
47      for staff in staffs:
48          subject_ids =
    Subjects.objects.filter(staff_id=staff.admin.id)
49          attendance =
    Attendance.objects.filter(subject_id__in=subject_ids).coun
    t()
50          leaves =
    LeaveReportStaff.objects.filter(staff_id=staff.id,
51
    leave_status=1).count()
52          staff_attendance_present_list.append(attendance)
53          staff_attendance_leave_list.append(leaves)
54          staff_name_list.append(staff.admin.first_name)
55
56      # For Students
57      student_attendance_present_list=[]
58      student_attendance_leave_list=[]
59      student_name_list=[]
60
```

```
    students = Students.objects.all()
    for student in students:
        attendance = AttendanceReport.objects.filter(student_id=student.id, status=True).count()
        absent = AttendanceReport.objects.filter(student_id=student.id, status=False).count()
        leaves = LeaveReportStudent.objects.filter(student_id=student.id, leave_status=1).count()
        student_attendance_present_list.append(attendance)
        student_attendance_leave_list.append(leaves+absent)
        student_name_list.append(student.admin.first_name)


    context={
        "all_student_count": all_student_count,
        "subject_count": subject_count,
        "course_count": course_count,
        "staff_count": staff_count,
        "course_name_list": course_name_list,
        "subject_count_list": subject_count_list,
        "student_count_list_in_course": student_count_list_in_course,
        "subject_list": subject_list,
        "student_count_list_in_subject": student_count_list_in_subject,
        "staff_attendance_present_list": staff_attendance_present_list,
        "staff_attendance_leave_list": staff_attendance_leave_list,
        "staff_name_list": staff_name_list,
        "student_attendance_present_list": student_attendance_present_list,
        "student_attendance_leave_list": student_attendance_leave_list,
        "student_name_list": student_name_list,
    }
    return render(request, "hod_template/home_content.html", context)

```

```
94  def add_staff(request):
95      return render(request,
    "hod_template/add_staff_template.html")
96
97
98  def add_staff_save(request):
99      if request.method != "POST":
100         messages.error(request, "Invalid Method ")
101         return redirect('add_staff')
102     else:
103         first_name = request.POST.get('first_name')
104         last_name = request.POST.get('last_name')
105         username = request.POST.get('username')
106         email = request.POST.get('email')
107         password = request.POST.get('password')
108         address = request.POST.get('address')
109
110         try:
111             user =
    CustomUser.objects.create_user(username=username,
112
    password=password,
113
    email=email,
114
    first_name=first_name,
115
    last_name=last_name,
116
    user_type=2)
117             user.staffs.address = address
118             user.save()
119             messages.success(request, "Staff Added
    Successfully!")
120             return redirect('add_staff')
121         except:
122             messages.error(request, "Failed to Add
    Staff!")
123             return redirect('add_staff')
124
125
126
127 def manage_staff(request):
128     staffs = Staffs.objects.all()
```

```
        context = {
            "staffs": staffs
        }
        return render(request,
    "hod_template/manage_staff_template.html", context)


def edit_staff(request, staff_id):
        staff = Staffs.objects.get(admin=staff_id)

        context = {
            "staff": staff,
            "id": staff_id
        }
        return render(request,
    "hod_template/edit_staff_template.html", context)


def edit_staff_save(request):
        if request.method != "POST":
            return HttpResponse("<h2>Method Not Allowed</h2>")
        else:
            staff_id = request.POST.get('staff_id')
            username = request.POST.get('username')
            email = request.POST.get('email')
            first_name = request.POST.get('first_name')
            last_name = request.POST.get('last_name')
            address = request.POST.get('address')

            try:
                # INSERTING into Customuser Model
                user = CustomUser.objects.get(id=staff_id)
                user.first_name = first_name
                user.last_name = last_name
                user.email = email
                user.username = username
                user.save()

                # INSERTING into Staff Model
                staff_model =
    Staffs.objects.get(admin=staff_id)
                staff_model.address = address
                staff_model.save()

```

```
            messages.success(request, "Staff Updated
Successfully.")
171             return redirect('/edit_staff/'+staff_id)
172
173         except:
174             messages.error(request, "Failed to Update
Staff.")
175             return redirect('/edit_staff/'+staff_id)
176
177
178
179 def delete_staff(request, staff_id):
180     staff = Staffs.objects.get(admin=staff_id)
181     try:
182         staff.delete()
183         messages.success(request, "Staff Deleted
Successfully.")
184         return redirect('manage_staff')
185     except:
186         messages.error(request, "Failed to Delete Staff.")
187         return redirect('manage_staff')
188
189
190
191
192 def add_course(request):
193     return render(request,
"hod_template/add_course_template.html")
194
195
196 def add_course_save(request):
197     if request.method != "POST":
198         messages.error(request, "Invalid Method!")
199         return redirect('add_course')
200     else:
201         course = request.POST.get('course')
202         try:
203             course_model = Courses(course_name=course)
204             course_model.save()
205             messages.success(request, "Course Added
Successfully!")
206             return redirect('add_course')
207         except:
```

```
            messages.error(request, "Failed to Add Course!")
            return redirect('add_course')


def manage_course(request):
    courses = Courses.objects.all()
    context = {
        "courses": courses
    }
    return render(request, 'hod_template/manage_course_template.html', context)


def edit_course(request, course_id):
    course = Courses.objects.get(id=course_id)
    context = {
        "course": course,
        "id": course_id
    }
    return render(request, 'hod_template/edit_course_template.html', context)


def edit_course_save(request):
    if request.method != "POST":
        HttpResponse("Invalid Method")
    else:
        course_id = request.POST.get('course_id')
        course_name = request.POST.get('course')

        try:
            course = Courses.objects.get(id=course_id)
            course.course_name = course_name
            course.save()

            messages.success(request, "Course Updated Successfully.")
            return redirect('/edit_course/'+course_id)

        except:
            messages.error(request, "Failed to Update Course.")
            return redirect('/edit_course/'+course_id)
```

```
248
249 def delete_course(request, course_id):
250     course = Courses.objects.get(id=course_id)
251     try:
252         course.delete()
253         messages.success(request, "Course Deleted
    Successfully.")
254         return redirect('manage_course')
255     except:
256         messages.error(request, "Failed to Delete
    Course.")
257         return redirect('manage_course')
258
259
260 def manage_session(request):
261     session_years = SessionYearModel.objects.all()
262     context = {
263         "session_years": session_years
264     }
265     return render(request,
    "hod_template/manage_session_template.html", context)
266
267
268 def add_session(request):
269     return render(request,
    "hod_template/add_session_template.html")
270
271
272 def add_session_save(request):
273     if request.method != "POST":
274         messages.error(request, "Invalid Method")
275         return redirect('add_course')
276     else:
277         session_start_year =
    request.POST.get('session_start_year')
278         session_end_year =
    request.POST.get('session_end_year')
279
280         try:
281             sessionyear =
    SessionYearModel(session_start_year=session_start_year,
282
    session_end_year=session_end_year)
283             sessionyear.save()
```

```
            messages.success(request, "Session Year added
Successfully!")
285             return redirect("add_session")
286         except:
287             messages.error(request, "Failed to Add Session
Year")
288             return redirect("add_session")
289
290
291 def edit_session(request, session_id):
292     session_year =
SessionYearModel.objects.get(id=session_id)
293     context = {
294         "session_year": session_year
295     }
296     return render(request,
"hod_template/edit_session_template.html", context)
297
298
299 def edit_session_save(request):
300     if request.method != "POST":
301         messages.error(request, "Invalid Method!")
302         return redirect('manage_session')
303     else:
304         session_id = request.POST.get('session_id')
305         session_start_year =
request.POST.get('session_start_year')
306         session_end_year =
request.POST.get('session_end_year')
307
308         try:
309             session_year =
SessionYearModel.objects.get(id=session_id)
310             session_year.session_start_year =
session_start_year
311             session_year.session_end_year =
session_end_year
312             session_year.save()
313
314             messages.success(request, "Session Year
Updated Successfully.")
315             return redirect('/edit_session/'+session_id)
316         except:
317             messages.error(request, "Failed to Update
Session Year.")
```

```
            return redirect('/edit_session/'+session_id)


def delete_session(request, session_id):
    session = SessionYearModel.objects.get(id=session_id)
    try:
        session.delete()
        messages.success(request, "Session Deleted Successfully.")
        return redirect('manage_session')
    except:
        messages.error(request, "Failed to Delete Session.")
        return redirect('manage_session')


def add_student(request):
    form = AddStudentForm()
    context = {
        "form": form
    }
    return render(request, 'hod_template/add_student_template.html', context)




def add_student_save(request):
    if request.method != "POST":
        messages.error(request, "Invalid Method")
        return redirect('add_student')
    else:
        form = AddStudentForm(request.POST, request.FILES)

        if form.is_valid():
            first_name = form.cleaned_data['first_name']
            last_name = form.cleaned_data['last_name']
            username = form.cleaned_data['username']
            email = form.cleaned_data['email']
            password = form.cleaned_data['password']
            address = form.cleaned_data['address']
            session_year_id = form.cleaned_data['session_year_id']
            course_id = form.cleaned_data['course_id']
```

```
                gender = form.cleaned_data['gender']
359
360
361             if len(request.FILES) != 0:
362                 profile_pic = request.FILES['profile_pic']
363                 fs = FileSystemStorage()
364                 filename = fs.save(profile_pic.name,
    profile_pic)
365                 profile_pic_url = fs.url(filename)
366             else:
367                 profile_pic_url = None
368
369
370             try:
371                 user =
    CustomUser.objects.create_user(username=username,
372
    password=password,
373
    email=email,
374
    first_name=first_name,
375
    last_name=last_name,
376
    user_type=3)
377                 user.students.address = address
378
379                 course_obj =
    Courses.objects.get(id=course_id)
380                 user.students.course_id = course_obj
381
382                 session_year_obj =
    SessionYearModel.objects.get(id=session_year_id)
383                 user.students.session_year_id =
    session_year_obj
384
385                 user.students.gender = gender
386                 user.students.profile_pic =
    profile_pic_url
387                 user.save()
388                 messages.success(request, "Student Added
    Successfully!")
389                 return redirect('add_student')
390             except:
```

```
                messages.error(request, "Failed to Add Student!")
                return redirect('add_student')
        else:
            return redirect('add_student')


def manage_student(request):
    students = Students.objects.all()
    context = {
        "students": students
    }
    return render(request, 'hod_template/manage_student_template.html', context)


def edit_student(request, student_id):

    # Adding Student ID into Session Variable
    request.session['student_id'] = student_id

    student = Students.objects.get(admin=student_id)
    form = EditStudentForm()

    # Filling the form with Data from Database
    form.fields['email'].initial = student.admin.email
    form.fields['username'].initial = student.admin.username
    form.fields['first_name'].initial = student.admin.first_name
    form.fields['last_name'].initial = student.admin.last_name
    form.fields['address'].initial = student.address
    form.fields['course_id'].initial = student.course_id.id
    form.fields['gender'].initial = student.gender
    form.fields['session_year_id'].initial = student.session_year_id.id

    context = {
        "id": student_id,
        "username": student.admin.username,
        "form": form
    }
```

```
    return render(request,
"hod_template/edit_student_template.html", context)


def edit_student_save(request):
    if request.method != "POST":
        return HttpResponse("Invalid Method!")
    else:
        student_id = request.session.get('student_id')
        if student_id == None:
            return redirect('/manage_student')

        form = EditStudentForm(request.POST,
request.FILES)
        if form.is_valid():
            email = form.cleaned_data['email']
            username = form.cleaned_data['username']
            first_name = form.cleaned_data['first_name']
            last_name = form.cleaned_data['last_name']
            address = form.cleaned_data['address']
            course_id = form.cleaned_data['course_id']
            gender = form.cleaned_data['gender']
            session_year_id =
form.cleaned_data['session_year_id']

            # Getting Profile Pic first
            # First Check whether the file is selected or
not
            # Upload only if file is selected
            if len(request.FILES) != 0:
                profile_pic = request.FILES['profile_pic']
                fs = FileSystemStorage()
                filename = fs.save(profile_pic.name,
profile_pic)
                profile_pic_url = fs.url(filename)
            else:
                profile_pic_url = None

            try:
                # First Update into Custom User Model
                user =
CustomUser.objects.get(id=student_id)
                user.first_name = first_name
                user.last_name = last_name
```

```
                user.email = email
467             user.username = username
468             user.save()
469
470             # Then Update Students Table
471             student_model =
    Students.objects.get(admin=student_id)
472             student_model.address = address
473
474             course = Courses.objects.get(id=course_id)
475             student_model.course_id = course
476
477             session_year_obj =
    SessionYearModel.objects.get(id=session_year_id)
478             student_model.session_year_id =
    session_year_obj
479
480             student_model.gender = gender
481             if profile_pic_url != None:
482                 student_model.profile_pic =
    profile_pic_url
483             student_model.save()
484             # Delete student_id SESSION after the data
    is updated
485             del request.session['student_id']
486
487             messages.success(request, "Student Updated
    Successfully!")
488             return
    redirect('/edit_student/'+student_id)
489         except:
490             messages.success(request, "Failed to
    Uupdate Student.")
491             return
    redirect('/edit_student/'+student_id)
492     else:
493         return redirect('/edit_student/'+student_id)
494
495
496 def delete_student(request, student_id):
497     student = Students.objects.get(admin=student_id)
498     try:
499         student.delete()
500         messages.success(request, "Student Deleted
    Successfully.")
```

```
            return redirect('manage_student')
    except:
            messages.error(request, "Failed to Delete Student.")
            return redirect('manage_student')


def add_subject(request):
    courses = Courses.objects.all()
    staffs = CustomUser.objects.filter(user_type='2')
    context = {
        "courses": courses,
        "staffs": staffs
    }
    return render(request, 'hod_template/add_subject_template.html', context)


def add_subject_save(request):
    if request.method != "POST":
        messages.error(request, "Method Not Allowed!")
        return redirect('add_subject')
    else:
        subject_name = request.POST.get('subject')

        course_id = request.POST.get('course')
        course = Courses.objects.get(id=course_id)

        staff_id = request.POST.get('staff')
        staff = CustomUser.objects.get(id=staff_id)

        try:
            subject = Subjects(subject_name=subject_name,
                               course_id=course,
                               staff_id=staff)
            subject.save()
            messages.success(request, "Subject Added Successfully!")
            return redirect('add_subject')
        except:
            messages.error(request, "Failed to Add Subject!")
            return redirect('add_subject')
```

```
542
543 def manage_subject(request):
544     subjects = Subjects.objects.all()
545     context = {
546         "subjects": subjects
547     }
548     return render(request,
    'hod_template/manage_subject_template.html', context)
549
550
551 def edit_subject(request, subject_id):
552     subject = Subjects.objects.get(id=subject_id)
553     courses = Courses.objects.all()
554     staffs = CustomUser.objects.filter(user_type='2')
555     context = {
556         "subject": subject,
557         "courses": courses,
558         "staffs": staffs,
559         "id": subject_id
560     }
561     return render(request,
    'hod_template/edit_subject_template.html', context)
562
563
564 def edit_subject_save(request):
565     if request.method != "POST":
566         HttpResponse("Invalid Method.")
567     else:
568         subject_id = request.POST.get('subject_id')
569         subject_name = request.POST.get('subject')
570         course_id = request.POST.get('course')
571         staff_id = request.POST.get('staff')
572
573         try:
574             subject = Subjects.objects.get(id=subject_id)
575             subject.subject_name = subject_name
576
577             course = Courses.objects.get(id=course_id)
578             subject.course_id = course
579
580             staff = CustomUser.objects.get(id=staff_id)
581             subject.staff_id = staff
582
```

```
            subject.save()
584
585             messages.success(request, "Subject Updated
    Successfully.")
586
587             return
    HttpResponseRedirect(reverse("edit_subject",
588                                                 kwargs=
    {"subject_id":subject_id}))
589
590         except:
591             messages.error(request, "Failed to Update
    Subject.")
592             return
    HttpResponseRedirect(reverse("edit_subject",
593                                                 kwargs=
    {"subject_id":subject_id}))
594
595
596
597
598 def delete_subject(request, subject_id):
599     subject = Subjects.objects.get(id=subject_id)
600     try:
601         subject.delete()
602         messages.success(request, "Subject Deleted
    Successfully.")
603         return redirect('manage_subject')
604     except:
605         messages.error(request, "Failed to Delete
    Subject.")
606         return redirect('manage_subject')
607
608
609 @csrf_exempt
610 def check_email_exist(request):
611     email = request.POST.get("email")
612     user_obj =
    CustomUser.objects.filter(email=email).exists()
613     if user_obj:
614         return HttpResponse(True)
615     else:
616         return HttpResponse(False)
617
618
```

```
@csrf_exempt
def check_username_exist(request):
    username = request.POST.get("username")
    user_obj = CustomUser.objects.filter(username=username).exists()
    if user_obj:
        return HttpResponse(True)
    else:
        return HttpResponse(False)



def student_feedback_message(request):
    feedbacks = FeedBackStudent.objects.all()
    context = {
        "feedbacks": feedbacks
    }
    return render(request, 'hod_template/student_feedback_template.html', context)


@csrf_exempt
def student_feedback_message_reply(request):
    feedback_id = request.POST.get('id')
    feedback_reply = request.POST.get('reply')

    try:
        feedback = FeedBackStudent.objects.get(id=feedback_id)
        feedback.feedback_reply = feedback_reply
        feedback.save()
        return HttpResponse("True")

    except:
        return HttpResponse("False")


def staff_feedback_message(request):
    feedbacks = FeedBackStaffs.objects.all()
    context = {
        "feedbacks": feedbacks
    }
    return render(request, 'hod_template/staff_feedback_template.html', context)
```

```
660
661 @csrf_exempt
662 def staff_feedback_message_reply(request):
663     feedback_id = request.POST.get('id')
664     feedback_reply = request.POST.get('reply')
665
666     try:
667         feedback =
    FeedBackStaffs.objects.get(id=feedback_id)
668         feedback.feedback_reply = feedback_reply
669         feedback.save()
670         return HttpResponse("True")
671
672     except:
673         return HttpResponse("False")
674
675
676 def student_leave_view(request):
677     leaves = LeaveReportStudent.objects.all()
678     context = {
679         "leaves": leaves
680     }
681     return render(request,
    'hod_template/student_leave_view.html', context)
682
683 def student_leave_approve(request, leave_id):
684     leave = LeaveReportStudent.objects.get(id=leave_id)
685     leave.leave_status = 1
686     leave.save()
687     return redirect('student_leave_view')
688
689
690 def student_leave_reject(request, leave_id):
691     leave = LeaveReportStudent.objects.get(id=leave_id)
692     leave.leave_status = 2
693     leave.save()
694     return redirect('student_leave_view')
695
696
697 def staff_leave_view(request):
698     leaves = LeaveReportStaff.objects.all()
699     context = {
700         "leaves": leaves
```

```
        }
    return render(request, 'hod_template/staff_leave_view.html', context)


def staff_leave_approve(request, leave_id):
    leave = LeaveReportStaff.objects.get(id=leave_id)
    leave.leave_status = 1
    leave.save()
    return redirect('staff_leave_view')


def staff_leave_reject(request, leave_id):
    leave = LeaveReportStaff.objects.get(id=leave_id)
    leave.leave_status = 2
    leave.save()
    return redirect('staff_leave_view')


def admin_view_attendance(request):
    subjects = Subjects.objects.all()
    session_years = SessionYearModel.objects.all()
    context = {
        "subjects": subjects,
        "session_years": session_years
    }
    return render(request, "hod_template/admin_view_attendance.html", context)


@csrf_exempt
def admin_get_attendance_dates(request):

    subject_id = request.POST.get("subject")
    session_year = request.POST.get("session_year_id")

    # Students enroll to Course, Course has Subjects
    # Getting all data from subject model based on subject_id
    subject_model = Subjects.objects.get(id=subject_id)

    session_model = SessionYearModel.objects.get(id=session_year)
```

```
    attendance = Attendance.objects.filter(subject_id=subject_model,
741 session_year_id=session_model)
742
743     # Only Passing Student Id and Student Name Only
744     list_data = []
745
746     for attendance_single in attendance:
747         data_small={"id":attendance_single.id,
748 "attendance_date":str(attendance_single.attendance_date),
749 "session_year_id":attendance_single.session_year_id.id}
750         list_data.append(data_small)
751
752     return JsonResponse(json.dumps(list_data),
753                         content_type="application/json",
754                         safe=False)
755
756
757 @csrf_exempt
758 def admin_get_attendance_student(request):
759
760     # Getting Values from Ajax POST 'Fetch Student'
761     attendance_date = request.POST.get('attendance_date')
762     attendance = Attendance.objects.get(id=attendance_date)
763
764     attendance_data = AttendanceReport.objects.filter(attendance_id=attendance)
765     # Only Passing Student Id and Student Name Only
766     list_data = []
767
768     for student in attendance_data:
769         data_small={"id":student.student_id.admin.id,
770 "name":student.student_id.admin.first_name+" "+student.student_id.admin.last_name,
771                     "status":student.status}
772         list_data.append(data_small)
773
774     return JsonResponse(json.dumps(list_data), content_type="application/json", safe=False)
775
```

```
def admin_profile(request):
    user = CustomUser.objects.get(id=request.user.id)

    context={
        "user": user
    }
    return render(request,
'hod_template/admin_profile.html', context)


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

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



def staff_profile(request):
    pass


def student_profile(requtest):
    pass
```

Step 12: Now add **models.py** to our project. It stores all the models that will be used in our project.

Python

```python
from django.contrib.auth.models import AbstractUser
from django.db import models
from django.db.models.signals import post_save
from django.dispatch import receiver



class SessionYearModel(models.Model):
    id = models.AutoField(primary_key=True)
    session_start_year = models.DateField()
    session_end_year = models.DateField()
    objects = models.Manager()



# Overriding the Default Django Auth
# User and adding One More Field (user_type)
class CustomUser(AbstractUser):
    HOD = '1'
    STAFF = '2'
    STUDENT = '3'

    EMAIL_TO_USER_TYPE_MAP = {
        'hod': HOD,
        'staff': STAFF,
        'student': STUDENT
    }

    user_type_data = ((HOD, "HOD"), (STAFF, "Staff"), (STUDENT, "Student"))
    user_type = models.CharField(default=1, choices=user_type_data, max_length=10)


class AdminHOD(models.Model):
```

```
    id = models.AutoField(primary_key=True)
    admin = models.OneToOneField(CustomUser, on_delete = models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    objects = models.Manager()


class Staffs(models.Model):
    id = models.AutoField(primary_key=True)
    admin = models.OneToOneField(CustomUser, on_delete = models.CASCADE)
    address = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    objects = models.Manager()



class Courses(models.Model):
    id = models.AutoField(primary_key=True)
    course_name = models.CharField(max_length=255)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    objects = models.Manager()



class Subjects(models.Model):
    id =models.AutoField(primary_key=True)
    subject_name = models.CharField(max_length=255)

    # need to give default course
    course_id = models.ForeignKey(Courses, on_delete=models.CASCADE, default=1)
    staff_id = models.ForeignKey(CustomUser, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    objects = models.Manager()



```

```
class Students(models.Model):
    id = models.AutoField(primary_key=True)
    admin = models.OneToOneField(CustomUser, on_delete = models.CASCADE)
    gender = models.CharField(max_length=50)
    profile_pic = models.FileField()
    address = models.TextField()
    course_id = models.ForeignKey(Courses, on_delete=models.DO_NOTHING, default=1)
    session_year_id = models.ForeignKey(SessionYearModel, null=True,

on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    objects = models.Manager()


class Attendance(models.Model):

    # Subject Attendance
    id = models.AutoField(primary_key=True)
    subject_id = models.ForeignKey(Subjects, on_delete=models.DO_NOTHING)
    attendance_date = models.DateField()
    session_year_id = models.ForeignKey(SessionYearModel, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    objects = models.Manager()


class AttendanceReport(models.Model):
    # Individual Student Attendance
    id = models.AutoField(primary_key=True)
    student_id = models.ForeignKey(Students, on_delete=models.DO_NOTHING)
    attendance_id = models.ForeignKey(Attendance, on_delete=models.CASCADE)
    status = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    objects = models.Manager()


```

```
class LeaveReportStudent(models.Model):
    id = models.AutoField(primary_key=True)
    student_id = models.ForeignKey(Students,
on_delete=models.CASCADE)
    leave_date = models.CharField(max_length=255)
    leave_message = models.TextField()
    leave_status = models.IntegerField(default=0)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    objects = models.Manager()


class LeaveReportStaff(models.Model):
    id = models.AutoField(primary_key=True)
    staff_id = models.ForeignKey(Staffs,
on_delete=models.CASCADE)
    leave_date = models.CharField(max_length=255)
    leave_message = models.TextField()
    leave_status = models.IntegerField(default=0)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    objects = models.Manager()


class FeedBackStudent(models.Model):
    id = models.AutoField(primary_key=True)
    student_id = models.ForeignKey(Students,
on_delete=models.CASCADE)
    feedback = models.TextField()
    feedback_reply = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    objects = models.Manager()


class FeedBackStaffs(models.Model):
    id = models.AutoField(primary_key=True)
    staff_id = models.ForeignKey(Staffs,
on_delete=models.CASCADE)
    feedback = models.TextField()
    feedback_reply = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    objects = models.Manager()
```

```
152
153
154 class NotificationStudent(models.Model):
155     id = models.AutoField(primary_key=True)
156     student_id = models.ForeignKey(Students,
    on_delete=models.CASCADE)
157     message = models.TextField()
158     created_at = models.DateTimeField(auto_now_add=True)
159     updated_at = models.DateTimeField(auto_now=True)
160     objects = models.Manager()
161
162
163 class NotificationStaffs(models.Model):
164     id = models.AutoField(primary_key=True)
165     stafff_id = models.ForeignKey(Staffs,
    on_delete=models.CASCADE)
166     message = models.TextField()
167     created_at = models.DateTimeField(auto_now_add=True)
168     updated_at = models.DateTimeField(auto_now=True)
169     objects = models.Manager()
170
171
172 class StudentResult(models.Model):
173     id = models.AutoField(primary_key=True)
174     student_id = models.ForeignKey(Students,
    on_delete=models.CASCADE)
175     subject_id = models.ForeignKey(Subjects,
    on_delete=models.CASCADE, default=1)
176     subject_exam_marks = models.FloatField(default=0)
177     subject_assignment_marks = models.FloatField(default=0)
178     created_at = models.DateTimeField(auto_now_add=True)
179     updated_at = models.DateTimeField(auto_now=True)
180     objects = models.Manager()
181
182
183 #Creating Django Signals
184 @receiver(post_save, sender=CustomUser)
185
186 # Now Creating a Function which will
187 # automatically insert data in HOD, Staff or Student
188 def create_user_profile(sender, instance, created,
    **kwargs):
189     # if Created is true (Means Data Inserted)
```

```
        if created:
191
192         # Check the user_type and insert the data in
    respective tables
193         if instance.user_type == 1:
194             AdminHOD.objects.create(admin=instance)
195         if instance.user_type == 2:
196             Staffs.objects.create(admin=instance)
197         if instance.user_type == 3:
198             Students.objects.create(admin=instance,
199
    course_id=Courses.objects.get(id=1),
200
    session_year_id=SessionYearModel.objects.get(id=1),
201                                     address="",
202                                     profile_pic="",
203                                     gender="")
204
205
206 @receiver(post_save, sender=CustomUser)
207 def save_user_profile(sender, instance, **kwargs):
208     if instance.user_type == 1:
209         instance.adminhod.save()
210     if instance.user_type == 2:
211         instance.staffs.save()
212     if instance.user_type == 3:
213         instance.students.save()
```

Step 13: Go to student_management_project -> setting and add **AUTH_USER_MODEL = 'student_management_app.CustomUser'**.

```
from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

# This way we are telling Django to use our custom model instead the default one.
AUTH_USER_MODEL = 'student_management_app.CustomUser'
```

Step 14: Now create **forms.py**

Python

```python
from django import forms
from .models import Courses, SessionYearModel


class DateInput(forms.DateInput):
    input_type = "date"


class AddStudentForm(forms.Form):
    email = forms.EmailField(label="Email",
                             max_length=50,
                             widget=forms.EmailInput(attrs={"class":"form-control"}))
    password = forms.CharField(label="Password",
                               max_length=50,
                               widget=forms.PasswordInput(attrs={"class":"form-control"}))
    first_name = forms.CharField(label="First Name",
                                 max_length=50,
                                 widget=forms.TextInput(attrs={"class":"form-control"}))
    last_name = forms.CharField(label="Last Name",
                                max_length=50,
                                widget=forms.TextInput(attrs={"class":"form-control"}))
    username = forms.CharField(label="Username",
                               max_length=50,
                               widget=forms.TextInput(attrs={"class":"form-control"}))
    address = forms.CharField(label="Address",
                              max_length=50,
                              widget=forms.TextInput(attrs={"class":"form-control"}))

    #For Displaying Courses
    try:
        courses = Courses.objects.all()
        course_list = []
        for course in courses:
            single_course = (course.id, course.course_name)
            course_list.append(single_course)
```

```
    except:
        print("here")
        course_list = []

    #For Displaying Session Years
    try:
        session_years = SessionYearModel.objects.all()
        session_year_list = []
        for session_year in session_years:
            single_session_year = (session_year.id, str(session_year.session_start_year)+" to "+str(session_year.session_end_year))
            session_year_list.append(single_session_year)

    except:
        session_year_list = []

    gender_list = (
        ('Male','Male'),
        ('Female','Female')
    )

    course_id = forms.ChoiceField(label="Course",
                                  choices=course_list,
                                  widget=forms.Select(attrs={"class":"form-control"}))
    gender = forms.ChoiceField(label="Gender",
                               choices=gender_list,
                               widget=forms.Select(attrs={"class":"form-control"}))
    session_year_id = forms.ChoiceField(label="Session Year",
                                        choices=session_year_list,
                                        widget=forms.Select(attrs={"class":"form-control"}))
    profile_pic = forms.FileField(label="Profile Pic",
                                  required=False,
                                  widget=forms.FileInput(attrs={"class":"form-control"}))



class EditStudentForm(forms.Form):
    email = forms.EmailField(label="Email",
```

```
                                max_length=50,
74                              widget=forms.EmailInput(attrs=
{"class":"form-control"}))
75      first_name = forms.CharField(label="First Name",
76                                   max_length=50,
77
widget=forms.TextInput(attrs={"class":"form-control"}))
78      last_name = forms.CharField(label="Last Name",
79                                  max_length=50,
80
widget=forms.TextInput(attrs={"class":"form-control"}))
81      username = forms.CharField(label="Username",
82                                 max_length=50,
83
widget=forms.TextInput(attrs={"class":"form-control"}))
84      address = forms.CharField(label="Address",
85                                max_length=50,
86                                widget=forms.TextInput(attrs=
{"class":"form-control"}))
87
88      # For Displaying Courses
89      try:
90          courses = Courses.objects.all()
91          course_list = []
92          for course in courses:
93              single_course = (course.id, course.course_name)
94              course_list.append(single_course)
95      except:
96          course_list = []
97
98      # For Displaying Session Years
99      try:
100         session_years = SessionYearModel.objects.all()
101         session_year_list = []
102         for session_year in session_years:
103             single_session_year = (session_year.id,
str(session_year.session_start_year)+" to
"+str(session_year.session_end_year))
104             session_year_list.append(single_session_year)
105
106     except:
107         session_year_list = []
108
109
```

```
        gender_list = (
            ('Male','Male'),
            ('Female','Female')
        )

        course_id = forms.ChoiceField(label="Course",
                                      choices=course_list,
    widget=forms.Select(attrs={"class":"form-control"}))
        gender = forms.ChoiceField(label="Gender",
                                   choices=gender_list,
                                   widget=forms.Select(attrs=
    {"class":"form-control"}))
        session_year_id = forms.ChoiceField(label="Session
    Year",
    choices=session_year_list,
    widget=forms.Select(attrs={"class":"form-control"}))
        profile_pic = forms.FileField(label="Profile Pic",
                                      required=False,
    widget=forms.FileInput(attrs={"class":"form-control"}))
```

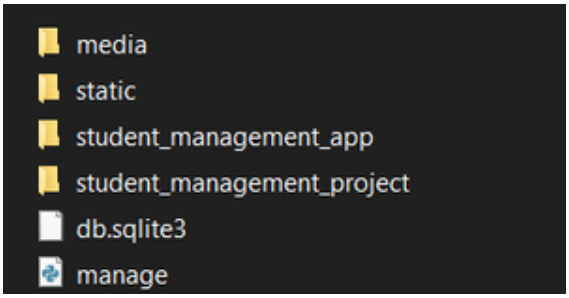
Step 15: Now register the models in **admin.py**

Python

```
from django.contrib import admin
from django.contrib.auth.admin import UserAdmin
from .models import CustomUser, AdminHOD, Staffs, Courses,
Subjects, Students, Attendance, AttendanceReport,
LeaveReportStudent, LeaveReportStaff, FeedBackStudent,
FeedBackStaffs, NotificationStudent, NotificationStaffs

# Register your models here.
class UserModel(UserAdmin):
    pass


admin.site.register(CustomUser, UserModel)

admin.site.register(AdminHOD)
```

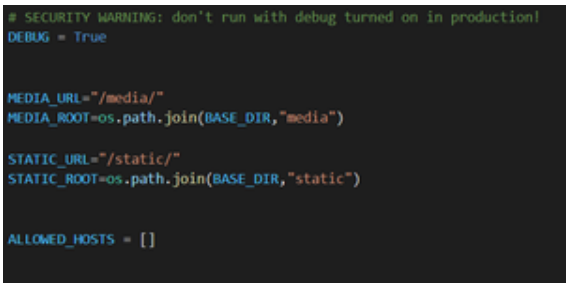
```
admin.site.register(Staffs)
admin.site.register(Courses)
admin.site.register(Subjects)
admin.site.register(Students)
admin.site.register(Attendance)
admin.site.register(AttendanceReport)
admin.site.register(LeaveReportStudent)
admin.site.register(LeaveReportStaff)
admin.site.register(FeedBackStudent)
admin.site.register(FeedBackStaffs)
admin.site.register(NotificationStudent)
admin.site.register(NotificationStaffs)
```

Step 16: Now Create a new folder as **templates** which includes Student_template, Hod_template, Staff_template folders. It contains the different templates used in each interface.
Create another folder as **static** which also includes some files. (Note – All these folders must be in student_management_project).



Step 17: Add media, static URLs, and root path.

```
import os

MEDIA_URL="/media/"
MEDIA_ROOT=os.path.join(BASE_DIR,"media")

STATIC_URL="/static/"
STATIC_ROOT=os.path.join(BASE_DIR,"static")
```



Step 18: Now create a **base.html** page.

HTML

```
{% load static %}

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>College Management System | Dashboard</title>
  <!-- Tell the browser to be responsive to screen width -->
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <!-- Font Awesome -->
  <link rel="stylesheet" href="{% static "fontawesome-free/css/all.min.css" %}">
  <!-- Ionicons -->

  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ckxlpbzKgwra6" crossorigin="anonymous">

  <link rel="stylesheet" href="https://code.ionicframework.com/ionicons/2.0.1/css/ionicons.min.css">
  <!-- Tempusdominus Bbootstrap 4 -->
  <link rel="stylesheet" href="{% static 'tempusdominus-
```

```
    bootstrap-4/css/tempusdominus-bootstrap-4.min.css' %}">
22    <!-- iCheck -->
23    <link rel="stylesheet" href="{% static "icheck-
    bootstrap/icheck-bootstrap.min.css" %}">
24    <!-- JQVMap -->
25    <link rel="stylesheet" href="{% static
    "jqvmap/jqvmap.min.css" %}">
26    <!-- Theme style -->
27    <link rel="stylesheet" href="{% static
    'dist/css/adminlte.min.css'  %}">
28    <!-- overlayScrollbars -->
29    <link rel="stylesheet" href="{% static
    "overlayScrollbars/css/OverlayScrollbars.min.css" %}">
30    <!-- Daterange picker -->
31    <link rel="stylesheet" href="{% static
    "daterangepicker/daterangepicker.css" %}">
32    <!-- summernote -->
33    <link rel="stylesheet" href="{% static
    "summernote/summernote-bs4.css" %}">
34    <!-- Google Font: Source Sans Pro -->
35    <link href="https://fonts.googleapis.com/css?
    family=Source+Sans+Pro:300,400,400i,700" rel="stylesheet">
36
37  </head>
38
39
40
41  {% block content %}
42
43
44  {% endblock content %}
45
46  <!-- jQuery -->
47      <!-- Optional JavaScript -->
48      <!-- jQuery first, then Popper.js, then Bootstrap JS -
    ->
49      <script src="https://code.jquery.com/jquery-
    3.4.1.slim.min.js" integrity="sha384-
    J6qa4849blE2+poT4WnyKhv5vZF5SrPo0iEjwBvKU7imGFAV0wwj1yYfoR
    SJoZ+n" crossorigin="anonymous"></script>
50      <script
    src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/um
    d/popper.min.js" integrity="sha384-
    Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvox
    MfooAo" crossorigin="anonymous"></script>
```

```
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js
/bootstrap.min.js" integrity="sha384-
wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og
8ifwB6" crossorigin="anonymous"></script>

<script src="{% static "jquery/jquery.min.js"  %}">
</script>
<!-- jQuery UI 1.11.4 -->
<script src="{% static "jquery-ui/jquery-ui.min.js"  %}">
</script>
<!-- Resolve conflict in jQuery UI tooltip with Bootstrap
tooltip -->
<script>
  $.widget.bridge('uibutton', $.ui.button)
</script>
<!-- Bootstrap 4 -->
<script src="{% static
"bootstrap/js/bootstrap.bundle.min.js"  %}"></script>
<!-- ChartJS -->
<script src="{% static "chart.js/Chart.min.js"  %}">
</script>
<!-- Sparkline -->
<script src="{% static "sparklines/sparkline.js"  %}">
</script>
<!-- JQVMap -->
<script src="{% static "jqvmap/jquery.vmap.min.js"  %}">
</script>
<script src="{% static "jqvmap/maps/jquery.vmap.usa.js"
%}"></script>
<!-- jQuery Knob Chart -->
<script src="{% static "jquery-knob/jquery.knob.min.js"
%}"></script>
<!-- daterangepicker -->
<script src="{% static "moment/moment.min.js"  %}">
</script>
<script src="{% static
"daterangepicker/daterangepicker.js"  %}"></script>
<!-- Tempusdominus Bootstrap 4 -->
<script src="{% static "tempusdominus-bootstrap-
4/js/tempusdominus-bootstrap-4.min.js"  %}"></script>
<!-- Summernote -->
<script src="{% static "summernote/summernote-bs4.min.js"
%}"></script>
<!-- overlayScrollbars -->
```

```
    <script src="{% static
    "overlayScrollbars/js/jquery.overlayScrollbars.min.js"
    %}"></script>
80  <!-- AdminLTE App -->
81  <script src="{% static 'dist/js/adminlte.js'  %}">
    </script>
82  <!-- AdminLTE dashboard demo (This is only for demo
    purposes) -->
83  <script src="{% static 'dist/js/pages/dashboard.js'  %}">
    </script>
84  <!-- AdminLTE for demo purposes -->
85  <script src="{% static 'dist/js/demo.js'  %}"></script>
86  </body>
87  </html>
```

Step 19: Now create a **home.html** page of our project in the student_management_app\templates folder.

HTML

```
1   {% extends 'base.html' %}
2   {% load static %}
3   {% block title %}Home{% endblock title %}
4
5   {% block content %}
6   <html>
7   <head>
8
9   <style>
10  img {
11    background-size: cover;
12  }
13  body {background-color: coral;}
14
15  </style>
16
17  <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
    alpha.6/css/bootstrap.min.css" integrity="sha384-
    rwoIResjU2yc3z8GV/NPeZWAv56rSmLldC3R/AZzGRnGxQQKnKkoFVhFQh
    NUwEyJ" crossorigin="anonymous">
18    <script src="https://code.jquery.com/jquery-
    3.1.1.slim.min.js" integrity="sha384-
```

```
    A7FZj7v+d/sdmMqp/nOQwliLvUsJfDHW+k9Omg/a/EheAdgtzNs3hpfag6
    Ed950n" crossorigin="anonymous"></script>
19    <script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-
    alpha.6/js/bootstrap.min.js" integrity="sha384-
    vBWWzlZJ8ea9aCX4pEW3rVHjgjt7zpkNpZk+02D9phzyeVkE+jo0ieGizq
    PLForn" crossorigin="anonymous"></script>
20
21   <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/
    bootstrap.min.css" integrity="sha384-
    Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dA
    iS6JXm" crossorigin="anonymous">
22  <script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bo
    otstrap.min.js" integrity="sha384-
    JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76
    PVCmYl" crossorigin="anonymous"></script>
23  <script src="https://code.jquery.com/jquery-
    3.2.1.slim.min.js" integrity="sha384-
    KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hX
    pG5KkN" crossorigin="anonymous"></script>
24  <script
    src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.12
    .9/umd/popper.min.js" integrity="sha384-
    ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusv
    fa0b4Q" crossorigin="anonymous"></script>
25
26  </head>
27
28  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
29      <a class="navbar-brand" href=""><h3>WELCOME TO
    CMS</h3></a>
30    <button class="navbar-toggler" type="button" data-
    toggle="collapse" data-target="#navbarSupportedContent"
    aria-controls="navbarSupportedContent" aria-
    expanded="false" aria-label="Toggle navigation">
31      <span class="navbar-toggler-icon"></span>
32    </button>
33
34    <div class="collapse navbar-collapse"
    id="navbarSupportedContent">
35      <ul class="navbar-nav mr-auto">
36      </ul>
37      <form class="form-inline my-2 my-lg-0">
38        <!--<input class="form-control mr-sm-2"
    type="search" placeholder="Search" aria-label="Search">-->
```

```
  <a href="/logi" class="btn btn-outline-success my-1 mx-2">Login</a>

       <!--<input class="form-control mr-sm-2" type="Register" placeholder="Register" aria-label="Register">-->
     <a href="/registration" class="btn btn-outline-success my-1 mx-2">Register</a>
        <a href="/contact" class="btn btn-outline-danger my-1 mx-2">Contact Us</a>
    </form>
  </div>
</nav>

<div id="carouselExampleIndicators" class="carousel slide" data-ride="carousel">
  <ol class="carousel-indicators">

  </ol>
  <div class="carousel-inner">
    <div class="carousel-item active">
      {% comment %} <img class="d-block w-100" src="https://images.unsplash.com/20/cambridge.JPG?ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&ixlib=rb-1.2.1&auto=format&fit=crop&w=1030&q=80" alt="First slide"> {% endcomment %}
      <img src="{% static 'dist/img/111.png' %}" class="d-block w-100 h-100 size-cover" alt="...">
    </div>
          <div class="carousel-item">
      {% comment %} <img class="d-block w-100" src="https://images.unsplash.com/photo-1541339907198-e08756dedf3f?ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&ixlib=rb-1.2.1&auto=format&fit=crop&w=1050&q=80" alt="Second slide"> {% endcomment %}
      <img src="{% static 'dist/img/re.png' %}" class="d-block w-100 h-100 size-cover " alt="...">
    </div>
          <div class="carousel-item">
      {% comment %} <img class="d-block w-100" src="https://images.unsplash.com/photo-1541339907198-e08756dedf3f?ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&ixlib=rb-1.2.1&auto=format&fit=crop&w=1050&q=80" alt="Second slide"> {% endcomment %}
```

```
      <img src="{% static 'dist/img/e.png' %}" class="d-
block w-100 h-100 size-cover " alt="...">
    </div>
    <div class="carousel-item">
      {% comment %} <img class="d-block w-100"
src="https://images.unsplash.com/photo-1541339907198-
e08756dedf3f?
ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&ixlib=rb
-1.2.1&auto=format&fit=crop&w=1050&q=80" alt="Second
slide"> {% endcomment %}
      <img src="{% static 'dist/img/22.png' %}" class="d-
block w-100 h-100 size-cover " alt="...">
    </div>
    <div class="carousel-item">
      {% comment %} <img class="d-block w-100"
src="https://images.unsplash.com/photo-1503676260728-
1c00da094a0b?
ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&ixlib=rb
-1.2.1&auto=format&fit=crop&w=1009&q=80" alt="Third
slide"> {% endcomment %}
      <img src="{% static 'dist/img/33.png' %}" class="d-
block w-100 h-100 size-cover" alt="...">
    </div>
  </div>
  <a class="carousel-control-prev"
href="#carouselExampleIndicators" role="button" data-
slide="prev">
    <span class="carousel-control-prev-icon" aria-
hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next"
href="#carouselExampleIndicators" role="button" data-
slide="next">
    <span class="carousel-control-next-icon" aria-
hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>

</html>

{% endblock content %}
```

Output :

Step 20: Now create a **registration.html** page where students, staff, HOD can register themselves.

HTML

```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Untitled</title>
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.1.3/css/bootstrap.min.css">
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/ionicons/2.0.1/css/ionicons.min.css">
   <link rel="stylesheet" href="assets/css/style.css">
   <style>
      body{
      height:1000px;
      background:#475d62;
      background-color: cover;
      font-family: sans-seriff;
```

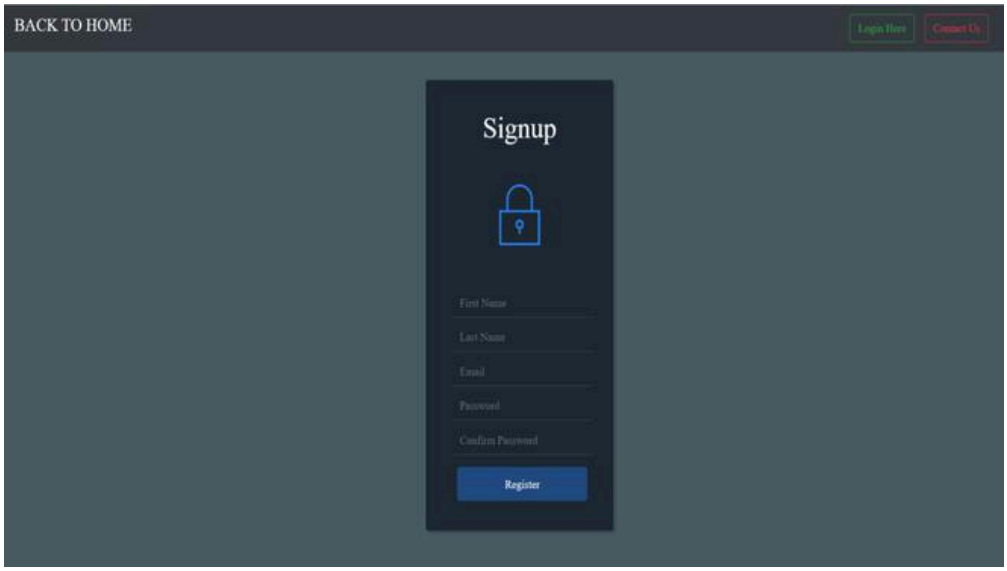
```
}
.login-dark {
max-width:320px;
width:90%;
background-color: #1e2833;
padding:40px;
border-radius:4px;
transform: translate(-50%,-50%);
position: absolute;
top: 50%;
left: 50%;
color:#fff;
box-shadow:3px 3px 4px rgba(0,0,0,0.2);
}
.login-dark form {
max-width:320px;
width:90%;
background-color:#1e2833;
padding:40px;
border-radius:4px;
transform:translate(-50%, -50%);
position:absolute;
top:50%;
left:50%;
color:#fff;
box-shadow:3px 3px 4px rgba(0,0,0,0.2);
}
.login-dark .illustration {
text-align:center;
padding:15px 0 20px;
font-size:100px;
color:#2980ef;
}
.login-dark form .form-control {
background:none;
border:none;
border-bottom:1px solid #434a52;
border-radius:0;
box-shadow:none;
outline:none;
color:inherit;
}
.login-dark form .btn-primary {
```

```
        background:#214a80;
        border:none;
        border-radius:4px;
        padding:11px;
        box-shadow:none;
        margin-top:26px;
        text-shadow:none;
        outline:none;
        }
        .login-dark form .btn-primary:hover, .login-dark form .btn-primary:active {
        background:#214a80;
        outline:none;
        }
        .login-dark form .forgot {
        display:block;
        text-align:center;
        font-size:12px;
        color:#6f7a85;
        opacity:0.9;
        text-decoration:none;
        }
        .login-dark form .forgot:hover, .login-dark form .forgot:active {
        opacity:1;
        text-decoration:none;
        }
        .login-dark form .btn-primary:active {
        transform:translateY(1px);
        }
    </style>
</head>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <a class="navbar-brand" href="/ ">
        <h4>BACK TO HOME</h4>
    </a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
```

```
        <ul class="navbar-nav mr-auto">
        </ul>
        <form class="form-inline my-2 my-lg-0">
            <!-- <input class="form-control mr-sm-2"
type="login" placeholder="login" aria-label="login">-->
            <!--<input class="form-control mr-sm-2"
type="Register" placeholder="Register" aria-
label="Register">-->
            <a href="/logi" class="btn btn-outline-success
my-1 mx-2">Login Here</a>
            <a href="/contact" class="btn btn-outline-danger
my-1 mx-2">Contact Us</a>
            <!--<input class="form-control mr-sm-2"
type="register" placeholder="register" aria-
label="register">-->
        </form>
    </div>
</nav>
<div class="login-dark form-inline py-0 mx-4 my-4 pl-4 pr-
4">
<form action="{% url 'doRegistration' %}" method="get">
{% csrf_token %}
<h1 class="text-center">Signup</h1>
<div class="illustration"><i class="icon ion-ios-locked-
outline"></i></div>
<div class="form-group"><input class="form-control mb-2"
type="text" name="first_name" placeholder="First Name">
</div>
<div class="form-group"><input class="form-control mb-2"
type="text" name="last_name" placeholder="Last Name">
</div>
<div class="form-group"><input class="form-control mb-2"
type="email" name="email" placeholder="Email"></div>
<div class="form-group"><input class="form-control mb-2"
type="password" name="password" placeholder="Password">
</div>
<div class="form-group"><input class="form-control mb-2"
type="password" name="confirmPassword"
placeholder="Confirm Password"></div>
<div class="form-group"><button class="btn btn-primary
btn-block mt-2 ml-2" type="submit">Register</button></div>
{% comment %} Display Messages {% endcomment %}
{% if messages %}
<div class="col-12">
    {% for message in messages %}
    {% if message.tags == "error" %}
```

```
        <div class="alert alert-danger alert-dismissible fade show" role="alert" style="margin-top: 10px;">
126         <b>{{ message }}</b>
127         <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
128     </div>
129     {% endif %}
130     {% endfor %}
131 </div>
132 {% endif %}
133 <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
134 <script src="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.1.3/js/bootstrap.bundle.min.js"></script>
135 {% endblock content %}
```

Output:



REGISTRATION PAGE

Step 21: Now create a **login_page.html** where students, staff, HOD can log in themselves.

HTML

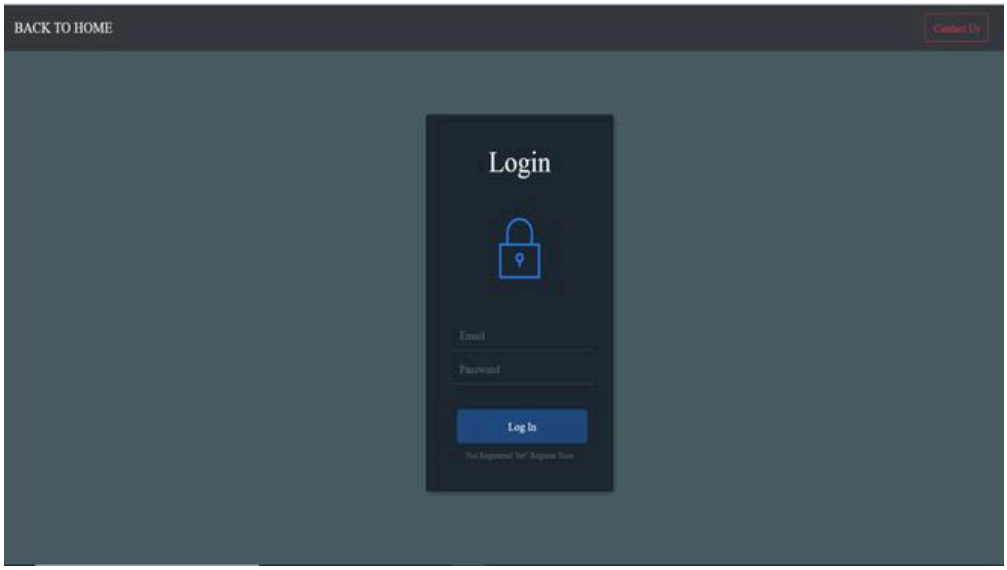
```
{% extends 'base.html' %}
{% load static %}
{% block content %}
<head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1; maximum-scale=1.0; user-scalable=0;">
   <title>Untitled</title>
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/4.1.3/css/bootstrap.min.css">
   <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/ionicons/2.0.1/css/ionicons.min.css">
   <link rel="stylesheet" href="assets/css/style.css">
   <style>
      body{
      height:1000px;
      background:#475d62;
      background-color: cover;
      font-family: sans-seriff;
      }
      .login-dark {
      max-width:320px;
      width:90%;
      background-color: #1e2833;
      padding:40px;
      border-radius:4px;
      transform: translate(-50%,-50%);
      position: absolute;
      top: 50%;
      left: 50%;
      color:#fff;
      box-shadow:3px 3px 4px rgba(0,0,0,0.2);
      }
      .login-dark form {
      max-width:320px;
      width:90%;
      background-color:#1e2833;
      padding:40px;
      border-radius:4px;
```

```
        transform:translate(-50%, -50%);
        position:absolute;
        top:50%;
        left:50%;
        color:#fff;
        box-shadow:3px 3px 4px rgba(0,0,0,0.2);
        }
        .login-dark .illustration {
        text-align:center;
        padding:15px 0 20px;
        font-size:100px;
        color:#2980ef;
        }
        .login-dark form .form-control {
        background:none;
        border:none;
        border-bottom:1px solid #434a52;
        border-radius:0;
        box-shadow:none;
        outline:none;
        color:inherit;
        }
        .login-dark form .btn-primary {
        background:#214a80;
        border:none;
        border-radius:4px;
        padding:11px;
        box-shadow:none;
        margin-top:26px;
        text-shadow:none;
        outline:none;
        }
        .login-dark form .btn-primary:hover, .login-dark form
.btn-primary:active {
        background:#214a80;
        outline:none;
        }
        .login-dark form .forgot {
        display:block;
        text-align:center;
        font-size:12px;
        color:#6f7a85;
        opacity:0.9;
```

```
        text-decoration:none;
80      }
81      .login-dark form .forgot:hover, .login-dark form
    .forgot:active {
82      opacity:1;
83      text-decoration:none;
84      }
85      .login-dark form .btn-primary:active {
86      transform:translateY(1px);
87      }
88    </style>
89  </head>
90  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
91    <a class="navbar-brand" href="/ ">BACK TO HOME</a>
92    <button class="navbar-toggler" type="button" data-
    toggle="collapse" data-target="#navbarSupportedContent"
    aria-controls="navbarSupportedContent" aria-
    expanded="false" aria-label="Toggle navigation">
93    <span class="navbar-toggler-icon"></span>
94    </button>
95    <div class="collapse navbar-collapse"
    id="navbarSupportedContent">
96      <ul class="navbar-nav mr-auto">
97      </ul>
98      <form class="form-inline my-2 my-lg-0">
99        <a href="{% url 'contact' %}" class="btn btn-
    outline-danger my-1 mx-2">Contact Us</a>
100     </form>
101     <!--        <form class="form-inline my-2 my-lg-0">--
    >
102   </div>
103 </nav>
104 <div class="login-dark form-inline py-0 mx-4 my-4 pl-4 pr-
    4">
105   <form action="{% url 'doLogin' %}" method="get">
106     {% csrf_token %}
107     <h1 class="text-center">Login</h1>
108     <div class="illustration"><i class="icon ion-ios-
    locked-outline"></i></div>
109     <div class="form-group"><input class="form-control
    mb-2" type="email" name="email" placeholder="Email"></div>
110     <div class="form-group"><input class="form-control
    mb-2" type="password" name="password"
    placeholder="Password"></div>
```

```
        <div class="form-group"><button class="btn btn-
    primary btn-block mb-2 ml-2" type="submit">Log In</button>
    </div>
112         <a href="/registration" class="forgot">Not Registered
    Yet? Register Now</a>
113     </form>
114 </div>
115 {% comment %} Display Messages {% endcomment %}
116 {% if messages %}
117 <div class="col-12">
118     {% for message in messages %}
119     {% if message.tags == "error" %}
120     {% comment %}
121     <div class="alert alert-danger alert-dismissible fade
    show" role="alert" style="margin-top: 10px;">
122         <b>{{ message }}</b>
123         <button type="button" class="btn-close" data-bs-
    dismiss="alert" aria-label="Close"></button>
124     </div>
125     {% endcomment %}
126     <div class="alert alert-danger alert-dismissible fade
    show" role="alert">
127         <strong>Invalid Login Credentials!</strong>
128         <button type="button" class="close" data-
    dismiss="alert" aria-label="Close">
129         <span aria-hidden="true">&times;</span>
130         </button>
131     </div>
132     {% endif %}
133     {% endfor %}
134 </div>
135 {% endif %}
136 <script
    src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.2.1/jq
    uery.min.js"></script>
137 <script
    src="https://cdnjs.cloudflare.com/ajax/libs/twitter-
    bootstrap/4.1.3/js/bootstrap.bundle.min.js"></script>
138 {% endblock content %}
```

Output:



LOGIN page

Step 22: Create **contact.html**

HTML

```
{% extends 'base.html' %}
{% load static %}

{% block content %}


 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
                  <a href="/ " class="btn btn-outline-primary my-1 mx-2">Go Back To Home </a>
                    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
                      <span class="navbar-toggler-icon"></span>
                    </button>

                    <div class="collapse navbar-collapse" id="navbarSupportedContent">
                      <ul class="navbar-nav mr-auto">


                      </ul>
```

```
                        <form class="form-inline my-2 my-lg-
0">
                         <!-- <input class="form-control mr-
sm-2" type="login" placeholder="login" aria-
label="login">-->

       <!--<input class="form-control mr-sm-2"
type="Register" placeholder="Register" aria-
label="Register">-->

                           <!--<input class="form-control
mr-sm-2" type="register" placeholder="register" aria-
label="register">-->

                        </form>
                        <form class="form-inline my-2 my-lg-
0">
                    </div>
                </nav>
<div class="container-fluid px-0">

    <img src="{% static 'dist/img/contact.jpg' %}"
class="d-block w-100 mx-0" alt="..." height=450px
width=10px>
</div>
<div class="container">
    <h1 class="text-center my-3 display-2">
        <b>Contact Us</b>
    </h1>
<form action="/contact" method="post">
{% csrf_token %}
    <div class="mb-3 py-2">
  <label for="exampleFormControlInput1" class="form-
label"><b>Name</b></label>
  <input type="name" class="form-control"
id="exampleFormControlInput1" name = "name"
placeholder="Enter your Name">
</div>
        <div class="mb-3 py-2">
            <label for="exampleFormControlInput1"
class="form-label"><b>Email id</b></label>
  <input type="email" class="form-control"
id="exampleFormControlInpu2"name = "email"
placeholder="Enter your Email">
</div>
        <div class="mb-3 py-2">
```

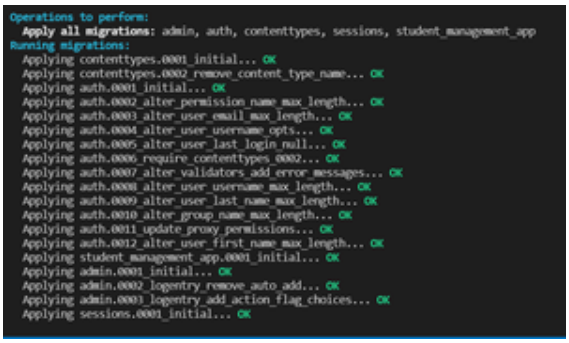
```
                <label for="exampleFormControlInput1"
    class="form-label"><b>Phone number</b></label>
49    <input type="number" class="form-control"
    id="exampleFormControlInput3"  name = "phone"
    placeholder="Enter your Phone number">
50  </div>
51  <div class="mb-3 py-2">
52      <label for="exampleFormControlTextarea1" class="form-
    label"><b>How can we help you ??</b></label>
53    <textarea class="form-control"
    id="exampleFormControlTextarea1" rows="7" name = "desc">
    </textarea>
54  </div>
55
56       <button type="submit" class="btn btn-primary btn-lg
    ">Submit</button>
57          </form>
58  </div>
59
60
61
62  {% endblock content%}
```

Step 23: Run these commands to migrate your models into the database. When you successfully do all the steps you will get this type of output in CMD.

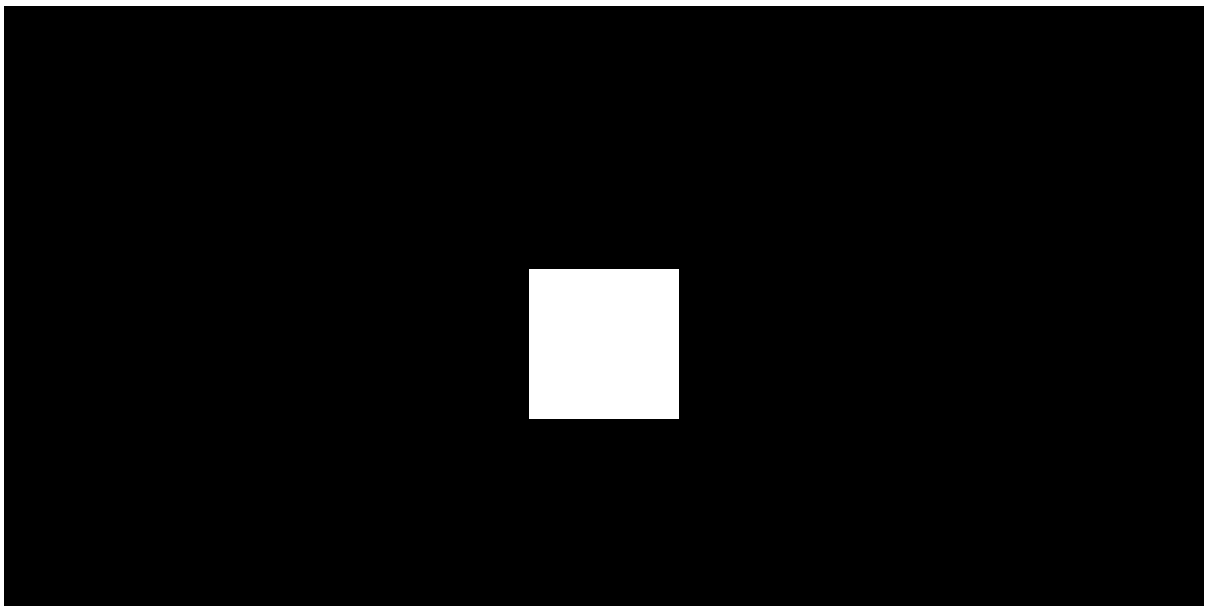
```
python manage.py makemigrations
```

```
student_management_app\migrations\0001_initial.py
  - Create model CustomUser
  - Create model Attendance
  - Create model Courses
  - Create model SessionYearModel
  - Create model Subjects
  - Create model Students
  - Create model StudentResult
  - Create model Staffs
  - Create model NotificationStudent
  - Create model NotificationStaffs
  - Create model LeaveReportStudent
  - Create model LeaveReportStaff
  - Create model FeedBackStudent
  - Create model FeedBackStaffs
  - Create model AttendanceReport
  - Add field session_year_id to attendance
  - Add field subject_id to attendance
  - Create model AdminHOD
```

```
python manage.py migrate
```



Project Overview:



00:00 04:57

Get complete project link here:

Github Link: click here

Are you ready to elevate your web development skills from foundational knowledge to advanced expertise? Explore our **Mastering Django Framework - Beginner to Advanced Course** on GeeksforGeeks, designed for aspiring developers and experienced programmers. This comprehensive course covers everything you need to know about Django, from the basics to advanced features. Gain practical experience through **hands-on projects** and real-world

applications, mastering essential Django principles and techniques. Whether you're just starting or looking to refine your skills, this course will empower you to build sophisticated web applications efficiently. Ready to enhance your web development journey? Enroll now and unlock your potential with Django!

N Nalin... 65

Previous Article

E-commerce Website using Django

Next Article

Create Word Counter app using Django

Similar Reads

Adding Tags Using Django-Taggit in Django Project

Django-Taggit is a Django application which is used to add tags to blogs, articles etc. It makes very easy for us to make adding the tags functionality to our djang...

2 min read

Event Management System Using Python Django

In today's fast-paced world, businesses, schools, and groups need to organize events well. An Event Management System (EMS) makes planning and managin...

13 min read

Voting System Project Using Django Framework

Project Title: Pollster (Voting System) web application using Django frameworkType of Application (Category): Web application. Introduction: We wil...

14 min read

Create Task Management System using Django

Task management systems are essential tools for individuals and organizations to organize, track, and prioritize tasks efficiently. In this article, we'll explore how to...

15+ min read

Django project to create a Comments System

Commenting on a post is the most common feature a post have and implementing in Django is way more easy than in other frameworks. To...

6 min read

Building Blog CMS (Content Management System) with Django

Django is a python based web application framework that is helpful for building a variety of web applications. Django also includes an extensible Django-Admin...

10 min read

Project Idea | (Project Approval System)

Academic Project management is a major issue which is faced by many educational institutes, the main reason for this is there is no automated system...

2 min read

Project Idea | College Connect

Project Title : College Connect College Connect is platform where everyone can share their pics or any other stuffs anonymously. Either it's a pdf file, images or a...

3 min read

Project Idea | College Network

Project Title College Network NOTE : All the feature provided below are not completed. Project is still under final phase of development. Introduction Colleg...

3 min read

Project Idea | Water Management System

Introduction: Water is one of the most important basic needs for living beings. But with the modernization and development of human lifestyles, consumption of...

5 min read

Article Tags : ProGeek Project Python Django-Projects +2 More

Practice Tags : python

Corporate & Communications Address:- A-143, 9th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305) | Registered Address:- K 061, Tower K, Gulshan Vivante Apartment, Sector 137, Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305