MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

IT Lab Mini Project Report on

Attendance Portal & Report Generation

SUBMITTED BY

Mahi Simhachal Reddy	46	210905252
S.V.Pramod	54	210905332
V.Lishitha	59	210905398

Section B

Under the Guidance of:
Mrs. Ancilla J Pinto and Mrs.Deepthi S
Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal, Karnataka – 576104

April 2024

TABLE OF CONTENTS

INTRODUCTION

In today's educational landscape, the effective management of student attendance is crucial for ensuring academic success and institutional efficiency. However, traditional methods of attendance tracking often prove cumbersome, time-consuming,
and prone to inaccuracies. To address these challenges and enhance the overall attendance management process, we introduce the Attendance Portal with Report Generation.The Attendance Portal with Report Generation is a comprehensive and innovative platform designed to streamline the recording, monitoring, and analysis of student attendance data. By harnessing the power of advanced technology and data analytics, the portal aims to revolutionize attendance management in higher education institutions.

PROBLEM STATEMENT

In higher education institutions, the manual process of tracking and managing student attendance poses significant challenges for administrators and educators.
Traditional methods, such as paper-based attendance registers or manual data entry
into spreadsheets, are time-consuming, error-prone, and lack real-time visibility. As a
result, there is a pressing need for a more efficient and automated solution to streamline attendance management processes.

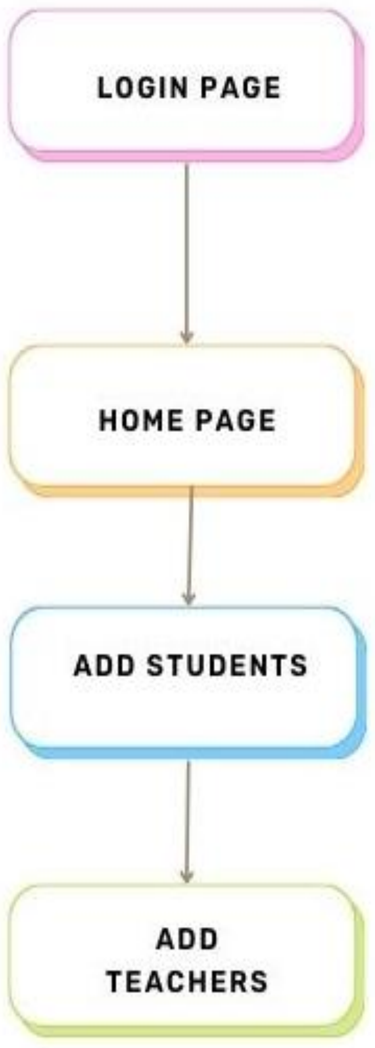
The successful development an implementation of such a system will not only streamline attendance management processes but also improve administrative efficiency, enhance student engagement, and ultimately contribute to the overall success of higher education institutions.

METHODOLOGY

Technology used: Django, Bootstrap, Html, Javascript, CSS

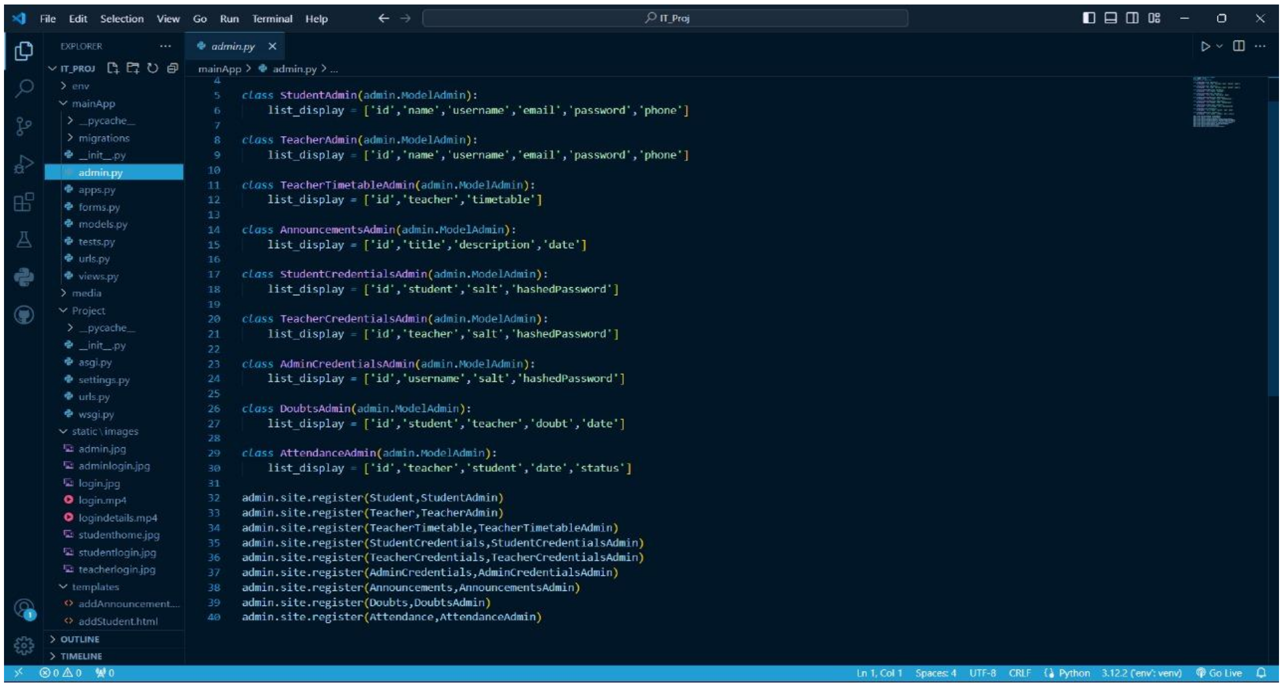
OS: Windows

Flowchart of the functionalities:

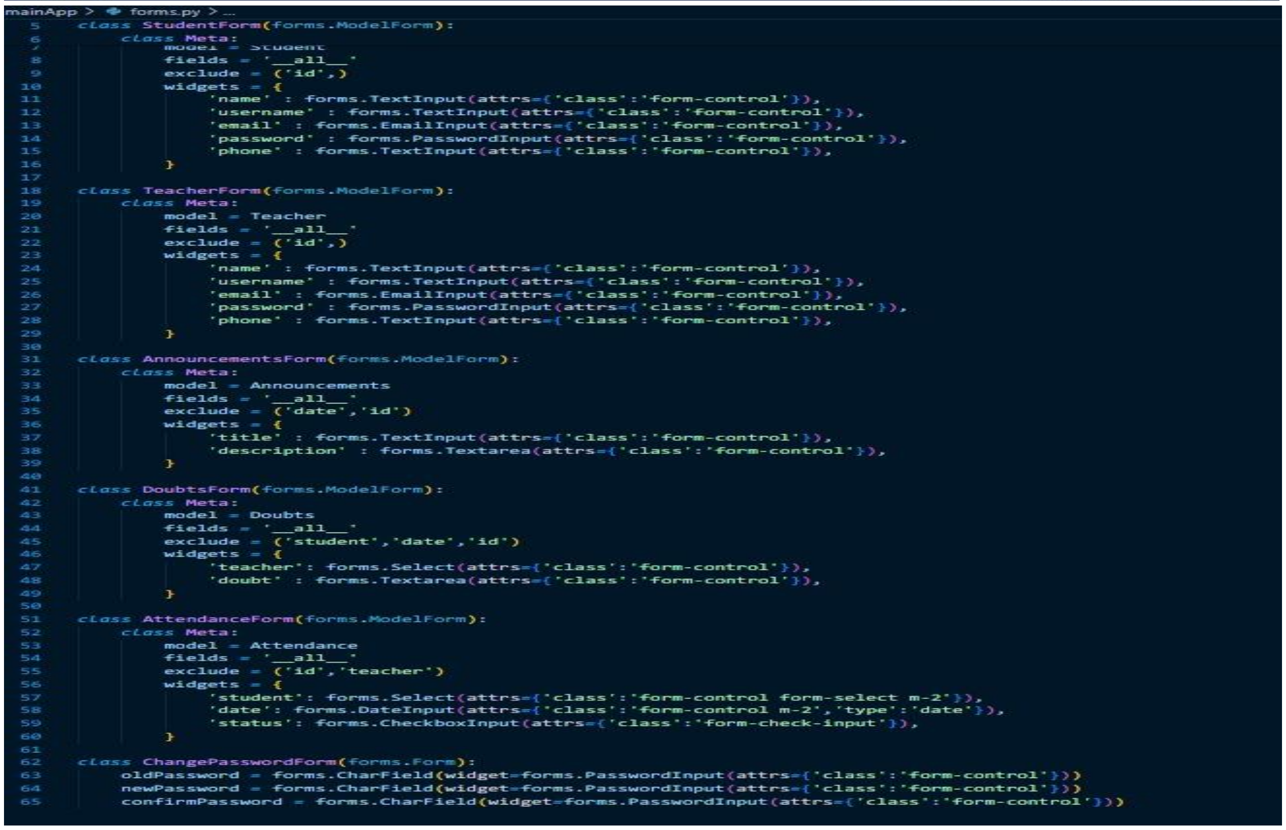


RESULTS AND SNAPSHOTS

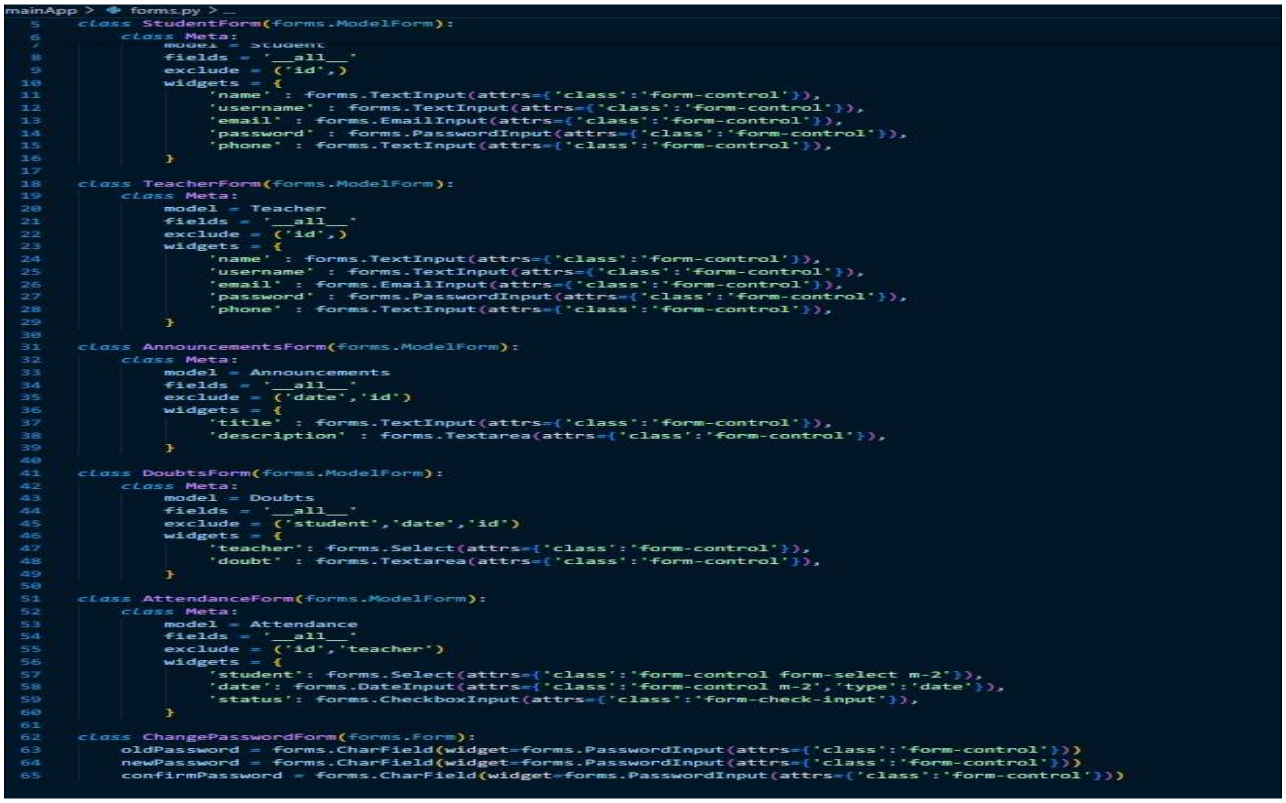
Code:



```python
class StudentAdmin(admin.ModelAdmin):
    list_display = ['id','name','username','email','password','phone']

class TeacherAdmin(admin.ModelAdmin):
    list_display = ['id','name','username','email','password','phone']

class TeacherTimetableAdmin(admin.ModelAdmin):
    list_display = ['id','teacher','timetable']

class AnnouncementsAdmin(admin.ModelAdmin):
    list_display = ['id','title','description','date']

class StudentCredentialsAdmin(admin.ModelAdmin):
    list_display = ['id','student','salt','hashedPassword']

class TeacherCredentialsAdmin(admin.ModelAdmin):
    list_display = ['id','teacher','salt','hashedPassword']

class AdminCredentialsAdmin(admin.ModelAdmin):
    list_display = ['id','username','salt','hashedPassword']

class DoubtsAdmin(admin.ModelAdmin):
    list_display = ['id','student','teacher','doubt','date']

class AttendanceAdmin(admin.ModelAdmin):
    list_display = ['id','teacher','student','date','status']

admin.site.register(Student,StudentAdmin)
admin.site.register(Teacher,TeacherAdmin)
admin.site.register(TeacherTimetable,TeacherTimetableAdmin)
admin.site.register(StudentCredentials,StudentCredentialsAdmin)
admin.site.register(TeacherCredentials,TeacherCredentialsAdmin)
admin.site.register(AdminCredentials,AdminCredentialsAdmin)
admin.site.register(Announcements,AnnouncementsAdmin)
admin.site.register(Doubts,DoubtsAdmin)
admin.site.register(Attendance,AttendanceAdmin)
```

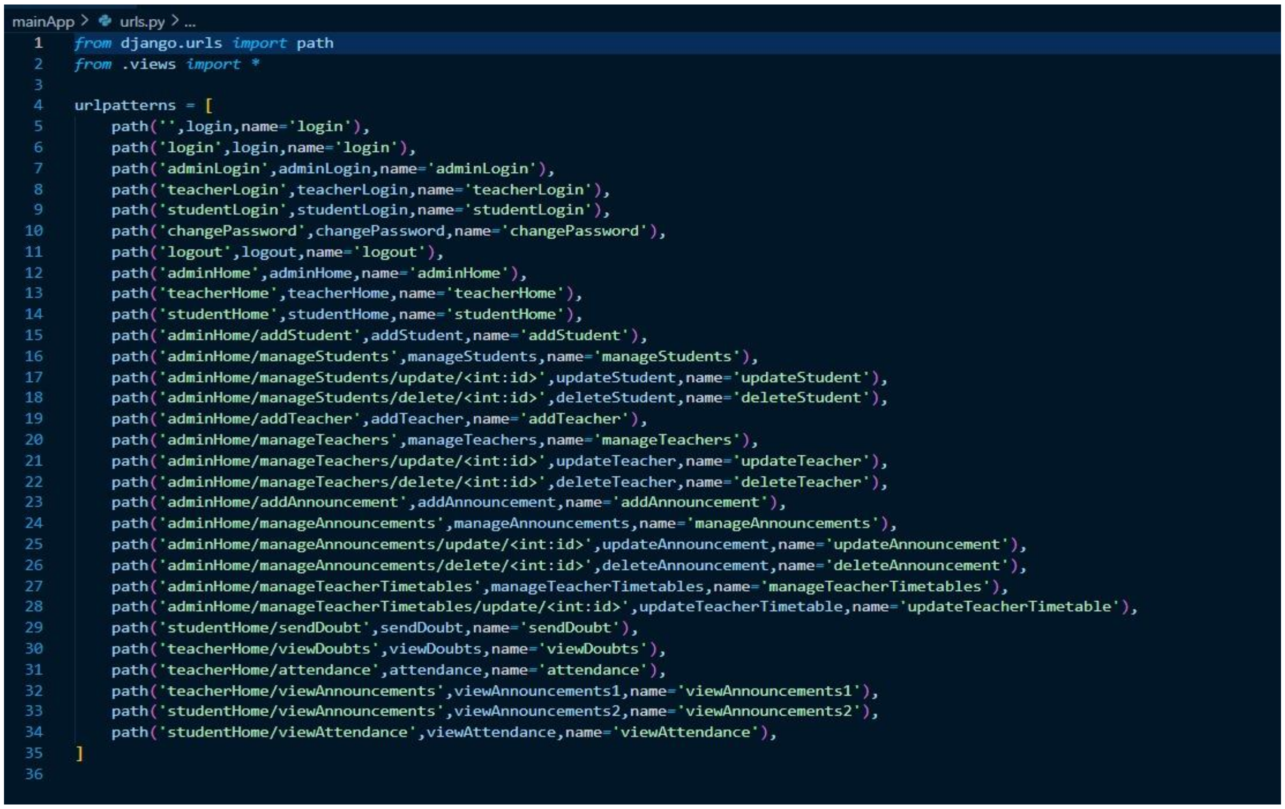


```python
class StudentForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = '__all__'
        exclude = ('id',)
        widgets = {
            'name' : forms.TextInput(attrs={'class':'form-control'}),
            'username' : forms.TextInput(attrs={'class':'form-control'}),
            'email' : forms.EmailInput(attrs={'class':'form-control'}),
            'password' : forms.PasswordInput(attrs={'class':'form-control'}),
            'phone' : forms.TextInput(attrs={'class':'form-control'}),
        }

class TeacherForm(forms.ModelForm):
    class Meta:
        model = Teacher
        fields = '__all__'
        exclude = ('id',)
        widgets = {
            'name' : forms.TextInput(attrs={'class':'form-control'}),
            'username' : forms.TextInput(attrs={'class':'form-control'}),
            'email' : forms.EmailInput(attrs={'class':'form-control'}),
            'password' : forms.PasswordInput(attrs={'class':'form-control'}),
            'phone' : forms.TextInput(attrs={'class':'form-control'}),
        }

class AnnouncementsForm(forms.ModelForm):
    class Meta:
        model = Announcements
        fields = '__all__'
        exclude = ('date','id')
        widgets = {
            'title' : forms.TextInput(attrs={'class':'form-control'}),
            'description' : forms.Textarea(attrs={'class':'form-control'}),
        }

class DoubtsForm(forms.ModelForm):
    class Meta:
        model = Doubts
        fields = '__all__'
        exclude = ('student','date','id')
        widgets = {
            'teacher': forms.Select(attrs={'class':'form-control'}),
            'doubt' : forms.Textarea(attrs={'class':'form-control'}),
        }

class AttendanceForm(forms.ModelForm):
    class Meta:
        model = Attendance
        fields = '__all__'
        exclude = ('id','teacher')
        widgets = {
            'student': forms.Select(attrs={'class':'form-control form-select m-2'}),
            'date': forms.DateInput(attrs={'class':'form-control m-2','type':'date'}),
            'status': forms.CheckboxInput(attrs={'class':'form-check-input'}),
        }

class ChangePasswordForm(forms.Form):
    oldPassword = forms.CharField(widget=forms.PasswordInput(attrs={'class':'form-control'}))
    newPassword = forms.CharField(widget=forms.PasswordInput(attrs={'class':'form-control'}))
    confirmPassword = forms.CharField(widget=forms.PasswordInput(attrs={'class':'form-control'}))
```



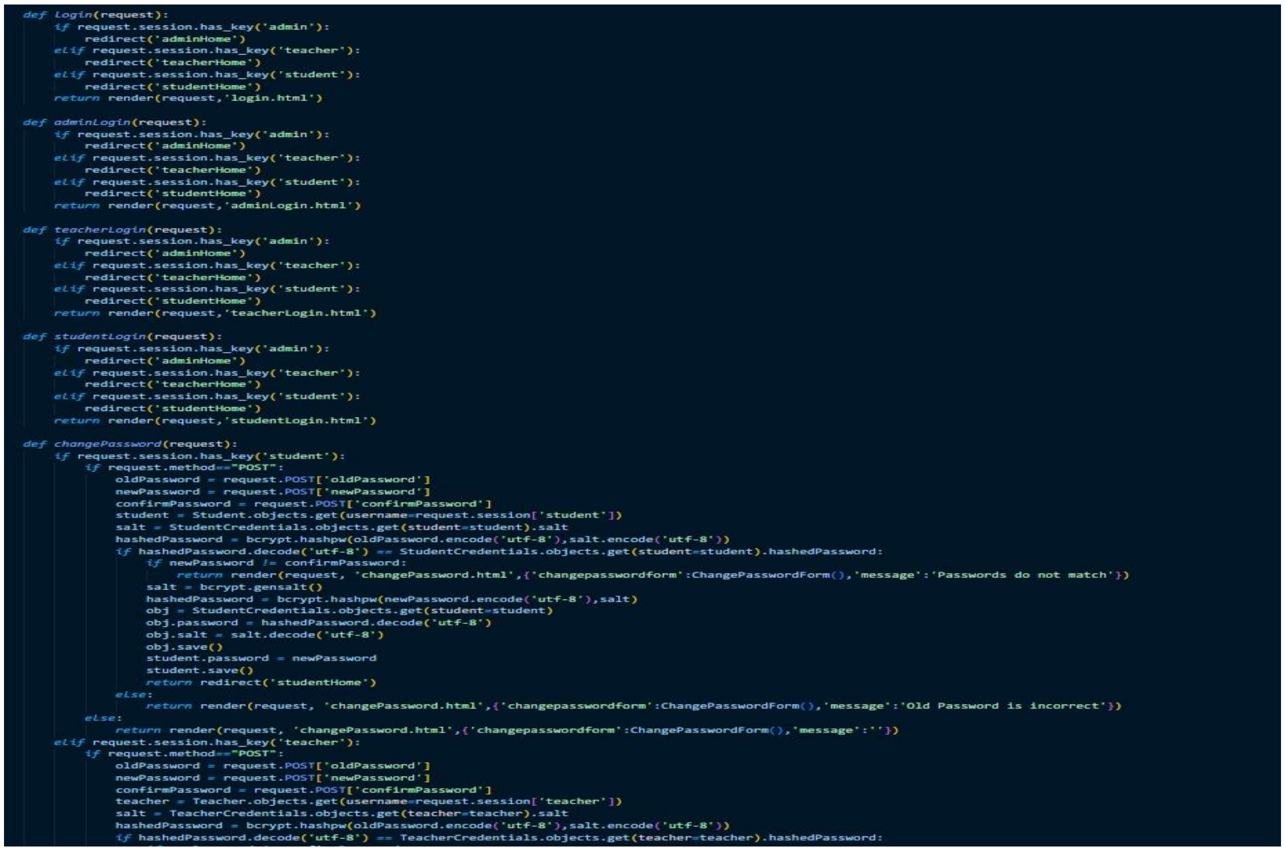
mainApp > forms.py > ...

```python
class StudentForm(forms.ModelForm):
    class Meta:
        model = Student
        fields = '__all__'
        exclude = ('id',)
        widgets = {
            'name' : forms.TextInput(attrs={'class':'form-control'}),
            'username' : forms.TextInput(attrs={'class':'form-control'}),
            'email' : forms.EmailInput(attrs={'class':'form-control'}),
            'password' : forms.PasswordInput(attrs={'class':'form-control'}),
            'phone' : forms.TextInput(attrs={'class':'form-control'}),
        }

class TeacherForm(forms.ModelForm):
    class Meta:
        model = Teacher
        fields = '__all__'
        exclude = ('id',)
        widgets = {
            'name' : forms.TextInput(attrs={'class':'form-control'}),
            'username' : forms.TextInput(attrs={'class':'form-control'}),
            'email' : forms.EmailInput(attrs={'class':'form-control'}),
            'password' : forms.PasswordInput(attrs={'class':'form-control'}),
            'phone' : forms.TextInput(attrs={'class':'form-control'}),
        }

class AnnouncementsForm(forms.ModelForm):
    class Meta:
        model = Announcements
        fields = '__all__'
        exclude = ('date','id')
        widgets = {
            'title' : forms.TextInput(attrs={'class':'form-control'}),
            'description' : forms.Textarea(attrs={'class':'form-control'}),
        }

class DoubtsForm(forms.ModelForm):
    class Meta:
        model = Doubts
        fields = '__all__'
        exclude = ('student','date','id')
        widgets = {
            'teacher': forms.Select(attrs={'class':'form-control'}),
            'doubt' : forms.Textarea(attrs={'class':'form-control'}),
        }

class AttendanceForm(forms.ModelForm):
    class Meta:
        model = Attendance
        fields = '__all__'
        exclude = ('id','teacher')
        widgets = {
            'student': forms.Select(attrs={'class':'form-control form-select m-2'}),
            'date': forms.DateInput(attrs={'class':'form-control m-2','type':'date'}),
            'status': forms.CheckboxInput(attrs={'class':'form-check-input'}),
        }

class ChangePasswordForm(forms.Form):
    oldPassword = forms.CharField(widget=forms.PasswordInput(attrs={'class':'form-control'}))
    newPassword = forms.CharField(widget=forms.PasswordInput(attrs={'class':'form-control'}))
    confirmPassword = forms.CharField(widget=forms.PasswordInput(attrs={'class':'form-control'}))
```

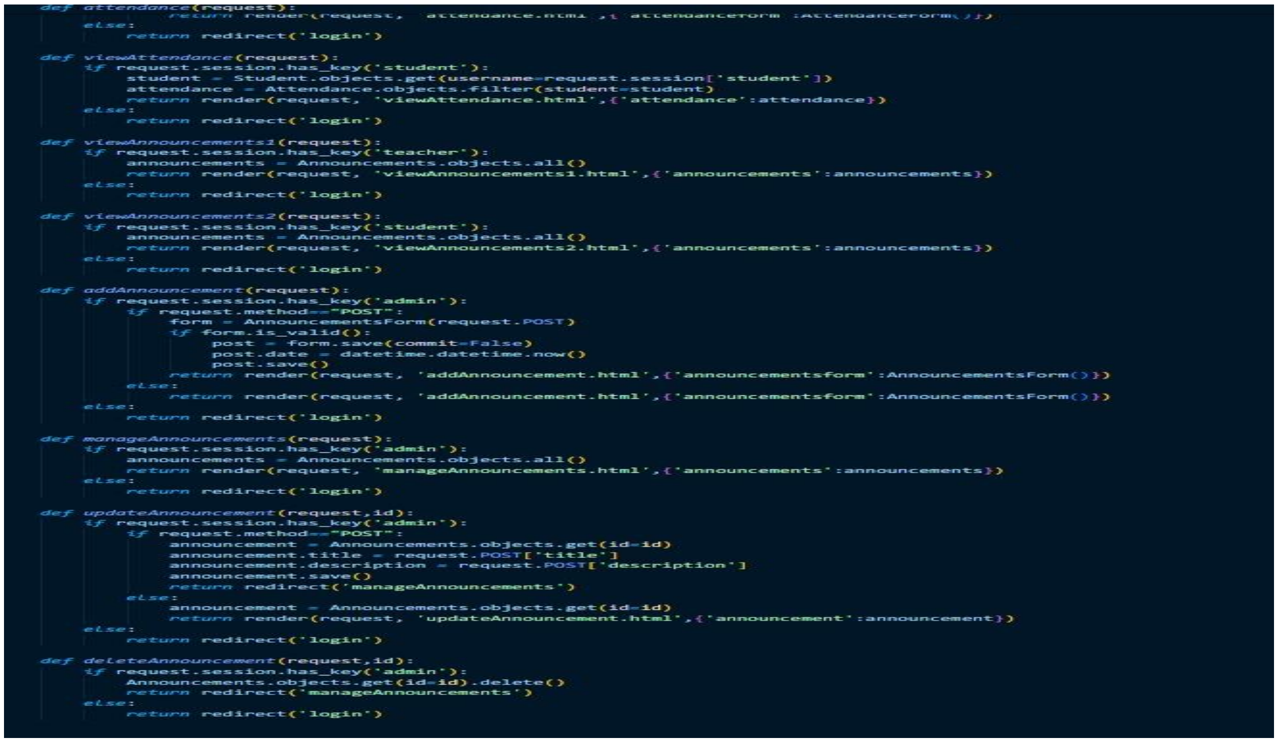


mainApp > urls.py > ...

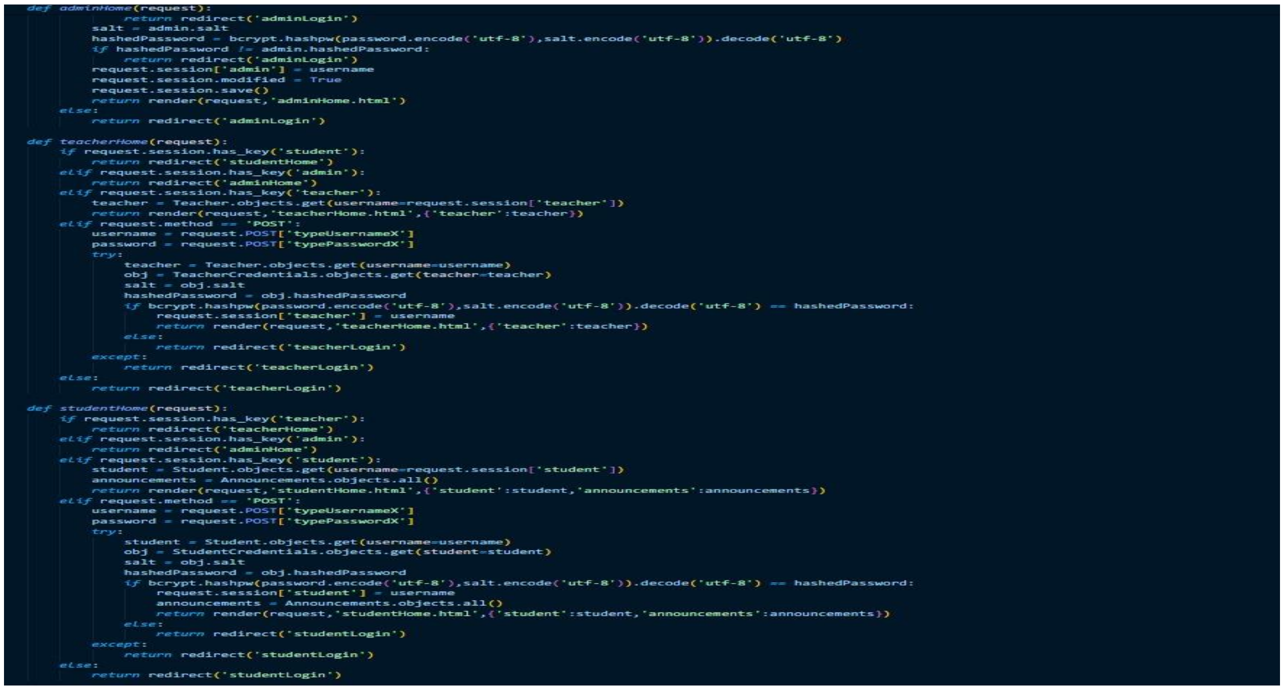
```python
from django.urls import path
from .views import *

urlpatterns = [
    path('',login,name='login'),
    path('login',login,name='login'),
    path('adminLogin',adminLogin,name='adminLogin'),
    path('teacherLogin',teacherLogin,name='teacherLogin'),
    path('studentLogin',studentLogin,name='studentLogin'),
    path('changePassword',changePassword,name='changePassword'),
    path('logout',logout,name='logout'),
    path('adminHome',adminHome,name='adminHome'),
    path('teacherHome',teacherHome,name='teacherHome'),
    path('studentHome',studentHome,name='studentHome'),
    path('adminHome/addStudent',addStudent,name='addStudent'),
    path('adminHome/manageStudents',manageStudents,name='manageStudents'),
    path('adminHome/manageStudents/update/<int:id>',updateStudent,name='updateStudent'),
    path('adminHome/manageStudents/delete/<int:id>',deleteStudent,name='deleteStudent'),
    path('adminHome/addTeacher',addTeacher,name='addTeacher'),
    path('adminHome/manageTeachers',manageTeachers,name='manageTeachers'),
    path('adminHome/manageTeachers/update/<int:id>',updateTeacher,name='updateTeacher'),
    path('adminHome/manageTeachers/delete/<int:id>',deleteTeacher,name='deleteTeacher'),
    path('adminHome/addAnnouncement',addAnnouncement,name='addAnnouncement'),
    path('adminHome/manageAnnouncements',manageAnnouncements,name='manageAnnouncements'),
    path('adminHome/manageAnnouncements/update/<int:id>',updateAnnouncement,name='updateAnnouncement'),
    path('adminHome/manageAnnouncements/delete/<int:id>',deleteAnnouncement,name='deleteAnnouncement'),
    path('adminHome/manageTeacherTimetables',manageTeacherTimetables,name='manageTeacherTimetables'),
    path('adminHome/manageTeacherTimetables/update/<int:id>',updateTeacherTimetable,name='updateTeacherTimetable'),
    path('studentHome/sendDoubt',sendDoubt,name='sendDoubt'),
    path('teacherHome/viewDoubts',viewDoubts,name='viewDoubts'),
    path('teacherHome/attendance',attendance,name='attendance'),
    path('teacherHome/viewAnnouncements',viewAnnouncements1,name='viewAnnouncements1'),
    path('studentHome/viewAnnouncements',viewAnnouncements2,name='viewAnnouncements2'),
    path('studentHome/viewAttendance',viewAttendance,name='viewAttendance'),
]
```



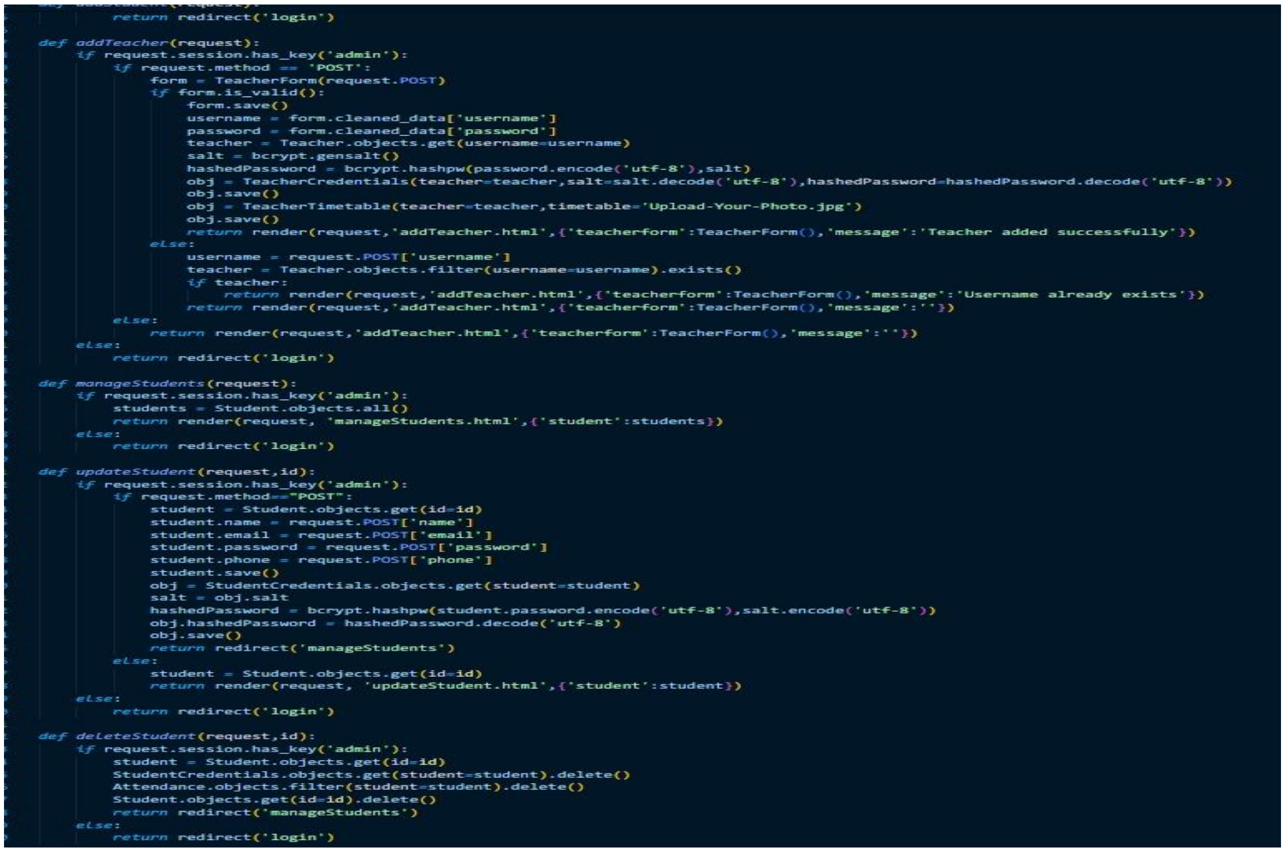
```python
def login(request):
    if request.session.has_key('admin'):
        redirect('adminHome')
    elif request.session.has_key('teacher'):
        redirect('teacherHome')
    elif request.session.has_key('student'):
        redirect('studentHome')
    return render(request,'login.html')

def adminLogin(request):
    if request.session.has_key('admin'):
        redirect('adminHome')
    elif request.session.has_key('teacher'):
        redirect('teacherHome')
    elif request.session.has_key('student'):
        redirect('studentHome')
    return render(request,'adminLogin.html')

def teacherLogin(request):
    if request.session.has_key('admin'):
        redirect('adminHome')
    elif request.session.has_key('teacher'):
        redirect('teacherHome')
    elif request.session.has_key('student'):
        redirect('studentHome')
    return render(request,'teacherLogin.html')

def studentLogin(request):
    if request.session.has_key('admin'):
        redirect('adminHome')
    elif request.session.has_key('teacher'):
        redirect('teacherHome')
    elif request.session.has_key('student'):
        redirect('studentHome')
    return render(request,'studentLogin.html')

def changePassword(request):
    if request.session.has_key('student'):
        if request.method=="POST":
            oldPassword = request.POST['oldPassword']
            newPassword = request.POST['newPassword']
            confirmPassword = request.POST['confirmPassword']
            student = Student.objects.get(username=request.session['student'])
            salt = StudentCredentials.objects.get(student=student).salt
            hashedPassword = bcrypt.hashpw(oldPassword.encode('utf-8'),salt.encode('utf-8'))
            if hashedPassword.decode('utf-8') == StudentCredentials.objects.get(student=student).hashedPassword:
                if newPassword != confirmPassword:
                    return render(request, 'changePassword.html',{'changepasswordform':ChangePasswordForm(),'message':'Passwords do not match'})
                salt = bcrypt.gensalt()
                hashedPassword = bcrypt.hashpw(newPassword.encode('utf-8'),salt)
                obj = StudentCredentials.objects.get(student=student)
                obj.password = hashedPassword.decode('utf-8')
                obj.salt = salt.decode('utf-8')
                obj.save()
                student.password = newPassword
                student.save()
                return redirect('studentHome')
            else:
                return render(request, 'changePassword.html',{'changepasswordform':ChangePasswordForm(),'message':'Old Password is incorrect'})
        else:
            return render(request, 'changePassword.html',{'changepasswordform':ChangePasswordForm(),'message':''})
    elif request.session.has_key('teacher'):
        if request.method=="POST":
            oldPassword = request.POST['oldPassword']
            newPassword = request.POST['newPassword']
            confirmPassword = request.POST['confirmPassword']
            teacher = Teacher.objects.get(username=request.session['teacher'])
            salt = TeacherCredentials.objects.get(teacher=teacher).salt
            hashedPassword = bcrypt.hashpw(oldPassword.encode('utf-8'),salt.encode('utf-8'))
            if hashedPassword.decode('utf-8') == TeacherCredentials.objects.get(teacher=teacher).hashedPassword:
```



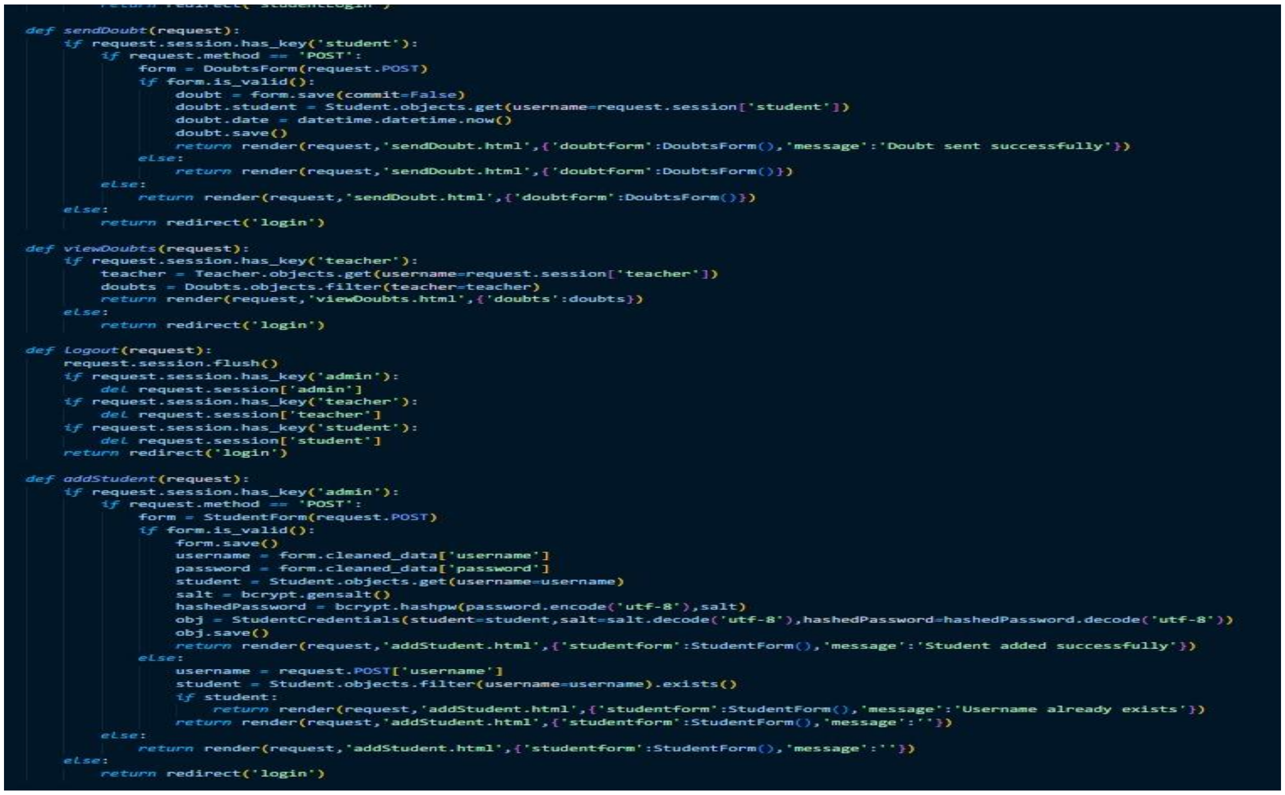
```python
def attendance(request):
            return render(request, 'attendance.html',{'attendanceform':AttendanceForm()})
    else:
        return redirect('login')

def viewAttendance(request):
    if request.session.has_key('student'):
        student = Student.objects.get(username=request.session['student'])
        attendance = Attendance.objects.filter(student=student)
        return render(request, 'viewAttendance.html',{'attendance':attendance})
    else:
        return redirect('login')

def viewAnnouncements1(request):
    if request.session.has_key('teacher'):
        announcements = Announcements.objects.all()
        return render(request, 'viewAnnouncements1.html',{'announcements':announcements})
    else:
        return redirect('login')

def viewAnnouncements2(request):
    if request.session.has_key('student'):
        announcements = Announcements.objects.all()
        return render(request, 'viewAnnouncements2.html',{'announcements':announcements})
    else:
        return redirect('login')

def addAnnouncement(request):
    if request.session.has_key('admin'):
        if request.method=="POST":
            form = AnnouncementsForm(request.POST)
            if form.is_valid():
                post = form.save(commit=False)
                post.date = datetime.datetime.now()
                post.save()
            return render(request, 'addAnnouncement.html',{'announcementsform':AnnouncementsForm()})
        else:
            return render(request, 'addAnnouncement.html',{'announcementsform':AnnouncementsForm()})
    else:
        return redirect('login')

def manageAnnouncements(request):
    if request.session.has_key('admin'):
        announcements = Announcements.objects.all()
        return render(request, 'manageAnnouncements.html',{'announcements':announcements})
    else:
        return redirect('login')

def updateAnnouncement(request,id):
    if request.session.has_key('admin'):
        if request.method=="POST":
            announcement = Announcements.objects.get(id=id)
            announcement.title = request.POST['title']
            announcement.description = request.POST['description']
            announcement.save()
            return redirect('manageAnnouncements')
        else:
            announcement = Announcements.objects.get(id=id)
            return render(request, 'updateAnnouncement.html',{'announcement':announcement})
    else:
        return redirect('login')

def deleteAnnouncement(request,id):
    if request.session.has_key('admin'):
        Announcements.objects.get(id=id).delete()
        return redirect('manageAnnouncements')
    else:
        return redirect('login')
```



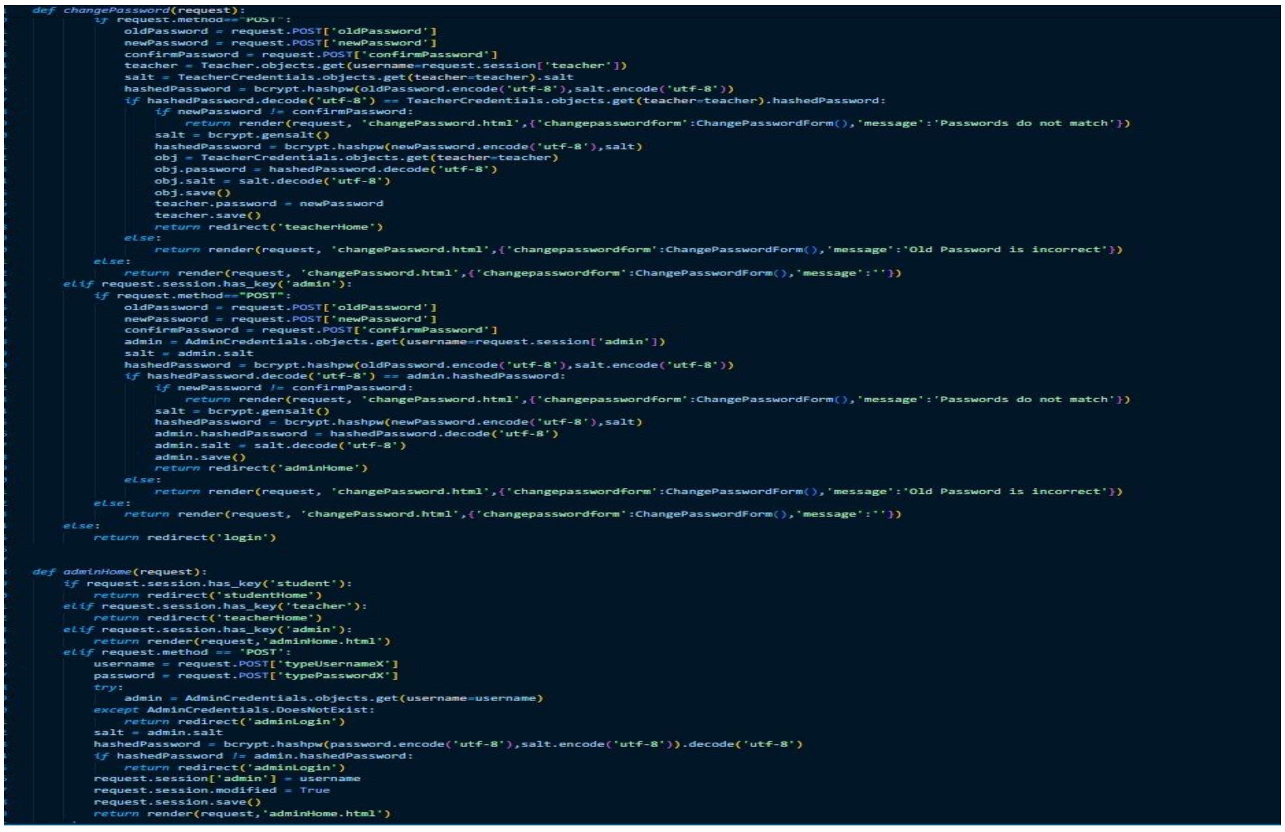
```python
def adminHome(request):
            return redirect('adminLogin')
        salt = admin.salt
        hashedPassword = bcrypt.hashpw(password.encode('utf-8'),salt.encode('utf-8')).decode('utf-8')
        if hashedPassword != admin.hashedPassword:
            return redirect('adminLogin')
        request.session['admin'] = username
        request.session.modified = True
        request.session.save()
        return render(request,'adminHome.html')
    else:
        return redirect('adminLogin')

def teacherHome(request):
    if request.session.has_key('student'):
        return redirect('studentHome')
    elif request.session.has_key('admin'):
        return redirect('adminHome')
    elif request.session.has_key('teacher'):
        teacher = Teacher.objects.get(username=request.session['teacher'])
        return render(request,'teacherHome.html',{'teacher':teacher})
    elif request.method == 'POST':
        username = request.POST['typeUsernameX']
        password = request.POST['typePasswordX']
        try:
            teacher = Teacher.objects.get(username=username)
            obj = TeacherCredentials.objects.get(teacher=teacher)
            salt = obj.salt
            hashedPassword = obj.hashedPassword
            if bcrypt.hashpw(password.encode('utf-8'),salt.encode('utf-8')).decode('utf-8') == hashedPassword:
                request.session['teacher'] = username
                return render(request,'teacherHome.html',{'teacher':teacher})
            else:
                return redirect('teacherLogin')
        except:
            return redirect('teacherLogin')
    else:
        return redirect('teacherLogin')

def studentHome(request):
    if request.session.has_key('teacher'):
        return redirect('teacherHome')
    elif request.session.has_key('admin'):
        return redirect('adminHome')
    elif request.session.has_key('student'):
        student = Student.objects.get(username=request.session['student'])
        announcements = Announcements.objects.all()
        return render(request,'studentHome.html',{'student':student,'announcements':announcements})
    elif request.method == 'POST':
        username = request.POST['typeUsernameX']
        password = request.POST['typePasswordX']
        try:
            student = Student.objects.get(username=username)
            obj = StudentCredentials.objects.get(student=student)
            salt = obj.salt
            hashedPassword = obj.hashedPassword
            if bcrypt.hashpw(password.encode('utf-8'),salt.encode('utf-8')).decode('utf-8') == hashedPassword:
                request.session['student'] = username
                announcements = Announcements.objects.all()
                return render(request,'studentHome.html',{'student':student,'announcements':announcements})
            else:
                return redirect('studentLogin')
        except:
            return redirect('studentLogin')
    else:
        return redirect('studentLogin')
```



```python
        return redirect('login')

def addTeacher(request):
    if request.session.has_key('admin'):
        if request.method == 'POST':
            form = TeacherForm(request.POST)
            if form.is_valid():
                form.save()
                username = form.cleaned_data['username']
                password = form.cleaned_data['password']
                teacher = Teacher.objects.get(username=username)
                salt = bcrypt.gensalt()
                hashedPassword = bcrypt.hashpw(password.encode('utf-8'),salt)
                obj = TeacherCredentials(teacher=teacher,salt=salt.decode('utf-8'),hashedPassword=hashedPassword.decode('utf-8'))
                obj.save()
                obj = TeacherTimetable(teacher=teacher,timetable='Upload-Your-Photo.jpg')
                obj.save()
                return render(request,'addTeacher.html',{'teacherform':TeacherForm(),'message':'Teacher added successfully'})
            else:
                username = request.POST['username']
                teacher = Teacher.objects.filter(username=username).exists()
                if teacher:
                    return render(request,'addTeacher.html',{'teacherform':TeacherForm(),'message':'Username already exists'})
                return render(request,'addTeacher.html',{'teacherform':TeacherForm(),'message':''})
        else:
            return render(request,'addTeacher.html',{'teacherform':TeacherForm(),'message':''})
    else:
        return redirect('login')

def manageStudents(request):
    if request.session.has_key('admin'):
        students = Student.objects.all()
        return render(request, 'manageStudents.html',{'student':students})
    else:
        return redirect('login')

def updateStudent(request,id):
    if request.session.has_key('admin'):
        if request.method=="POST":
            student = Student.objects.get(id=id)
            student.name = request.POST['name']
            student.email = request.POST['email']
            student.password = request.POST['password']
            student.phone = request.POST['phone']
            student.save()
            obj = StudentCredentials.objects.get(student=student)
            salt = obj.salt
            hashedPassword = bcrypt.hashpw(student.password.encode('utf-8'),salt.encode('utf-8'))
            obj.hashedPassword = hashedPassword.decode('utf-8')
            obj.save()
            return redirect('manageStudents')
        else:
            student = Student.objects.get(id=id)
            return render(request, 'updateStudent.html',{'student':student})
    else:
        return redirect('login')

def deleteStudent(request,id):
    if request.session.has_key('admin'):
        student = Student.objects.get(id=id)
        StudentCredentials.objects.get(student=student).delete()
        Attendance.objects.filter(student=student).delete()
        Student.objects.get(id=id).delete()
        return redirect('manageStudents')
    else:
        return redirect('login')
```

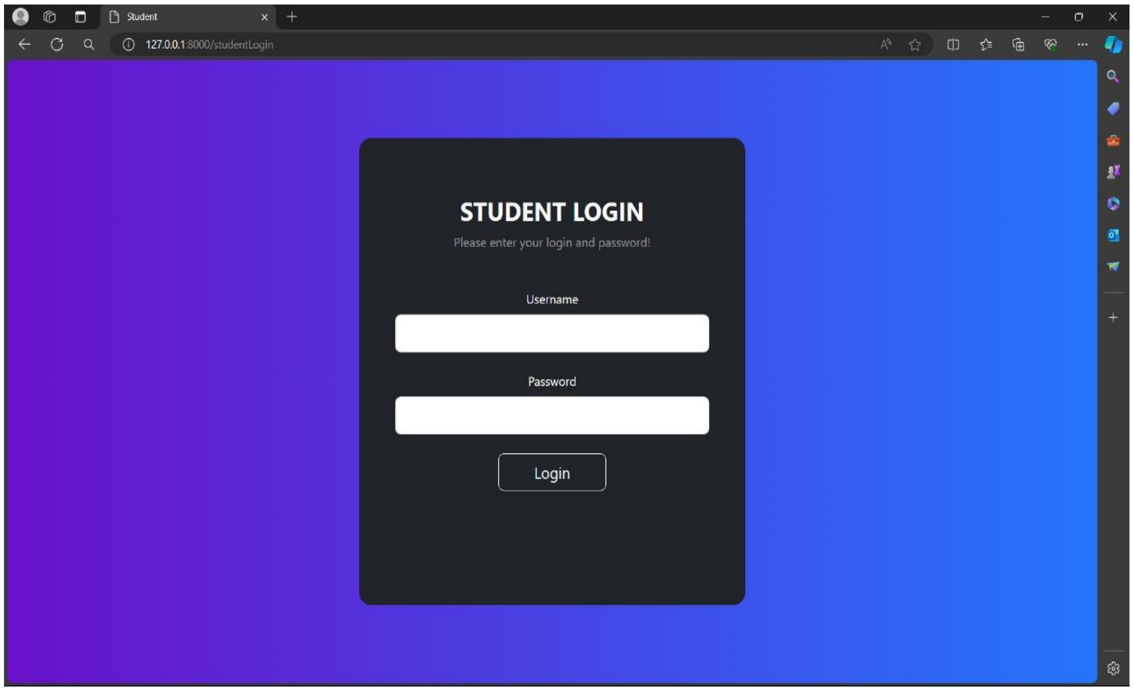


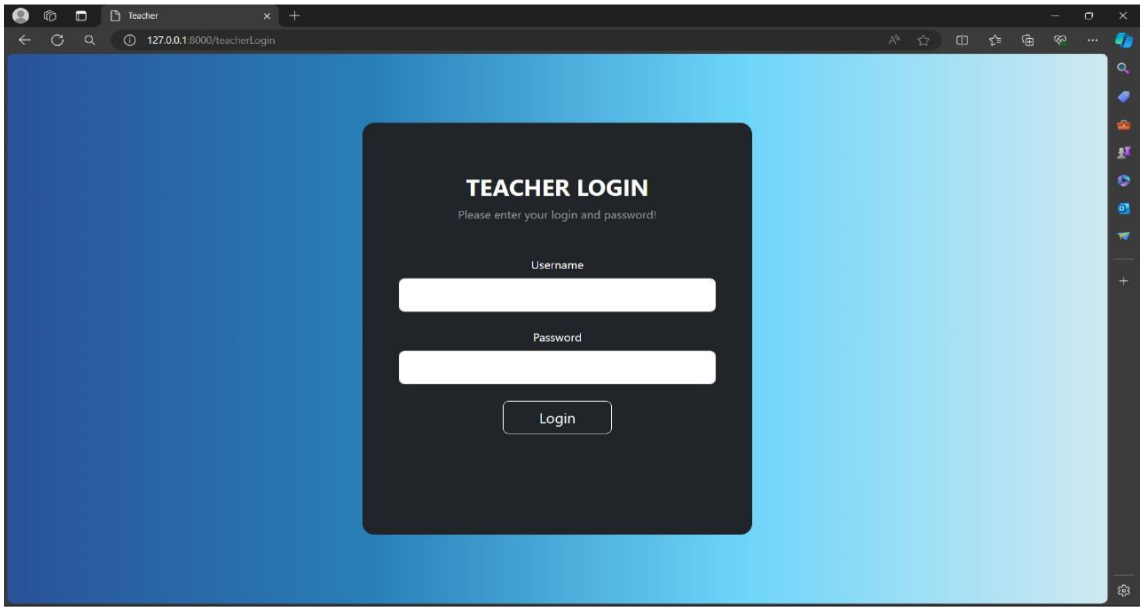
```python
            return redirect('studentLogin')

def sendDoubt(request):
    if request.session.has_key('student'):
        if request.method == 'POST':
            form = DoubtsForm(request.POST)
            if form.is_valid():
                doubt = form.save(commit=False)
                doubt.student = Student.objects.get(username=request.session['student'])
                doubt.date = datetime.datetime.now()
                doubt.save()
                return render(request,'sendDoubt.html',{'doubtform':DoubtsForm(),'message':'Doubt sent successfully'})
            else:
                return render(request,'sendDoubt.html',{'doubtform':DoubtsForm()})
        else:
            return render(request,'sendDoubt.html',{'doubtform':DoubtsForm()})
    else:
        return redirect('login')

def viewDoubts(request):
    if request.session.has_key('teacher'):
        teacher = Teacher.objects.get(username=request.session['teacher'])
        doubts = Doubts.objects.filter(teacher=teacher)
        return render(request,'viewDoubts.html',{'doubts':doubts})
    else:
        return redirect('login')

def logout(request):
    request.session.flush()
    if request.session.has_key('admin'):
        del request.session['admin']
    if request.session.has_key('teacher'):
        del request.session['teacher']
    if request.session.has_key('student'):
        del request.session['student']
    return redirect('login')

def addStudent(request):
    if request.session.has_key('admin'):
        if request.method == 'POST':
            form = StudentForm(request.POST)
            if form.is_valid():
                form.save()
                username = form.cleaned_data['username']
                password = form.cleaned_data['password']
                student = Student.objects.get(username=username)
                salt = bcrypt.gensalt()
                hashedPassword = bcrypt.hashpw(password.encode('utf-8'),salt)
                obj = StudentCredentials(student=student,salt=salt.decode('utf-8'),hashedPassword=hashedPassword.decode('utf-8'))
                obj.save()
                return render(request,'addStudent.html',{'studentform':StudentForm(),'message':'Student added successfully'})
            else:
                username = request.POST['username']
                student = Student.objects.filter(username=username).exists()
                if student:
                    return render(request,'addStudent.html',{'studentform':StudentForm(),'message':'Username already exists'})
                return render(request,'addStudent.html',{'studentform':StudentForm(),'message':''})
        else:
            return render(request,'addStudent.html',{'studentform':StudentForm(),'message':''})
    else:
        return redirect('login')
```

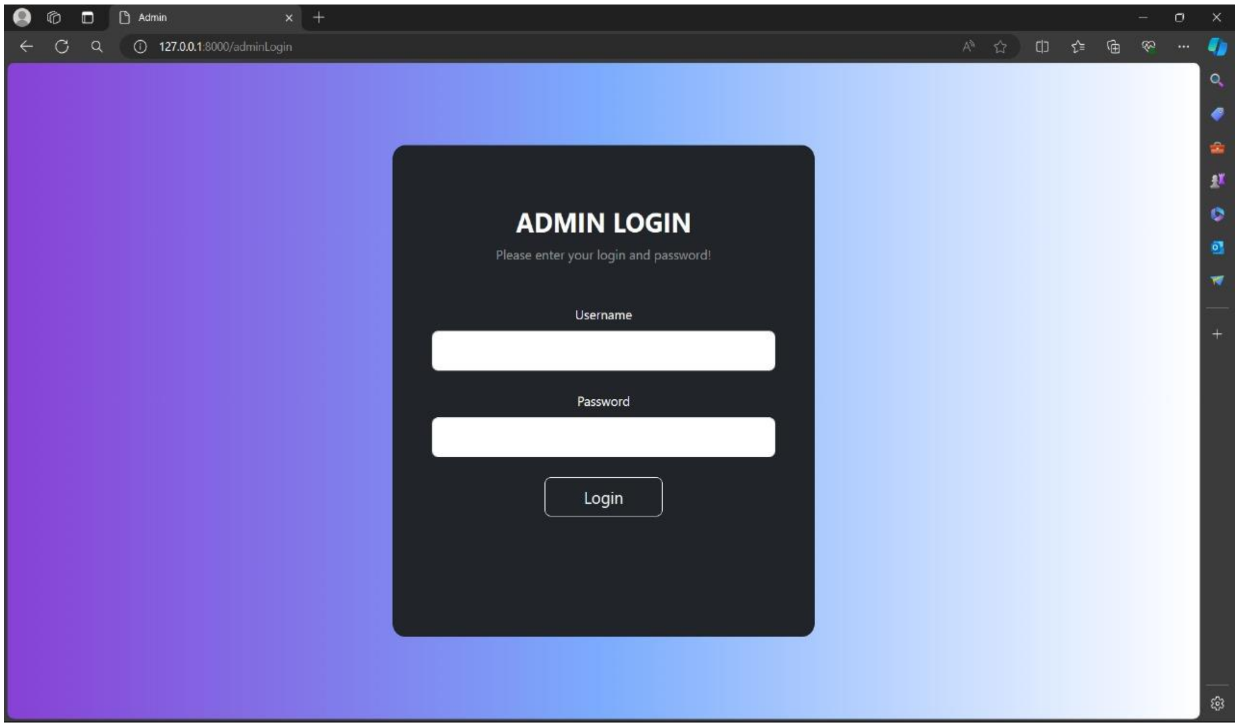


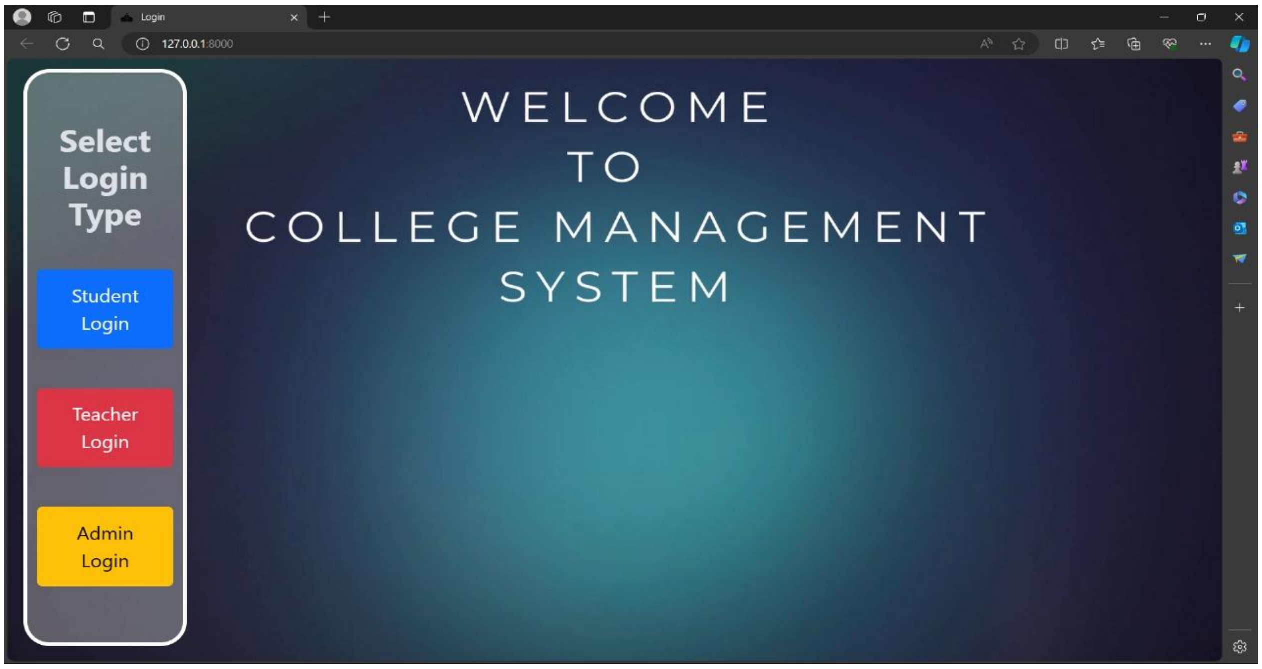
```python
def changePassword(request):
        if request.method=="POST":
            oldPassword = request.POST['oldPassword']
            newPassword = request.POST['newPassword']
            confirmPassword = request.POST['confirmPassword']
            teacher = Teacher.objects.get(username=request.session['teacher'])
            salt = TeacherCredentials.objects.get(teacher=teacher).salt
            hashedPassword = bcrypt.hashpw(oldPassword.encode('utf-8'),salt.encode('utf-8'))
            if hashedPassword.decode('utf-8') == TeacherCredentials.objects.get(teacher=teacher).hashedPassword:
                if newPassword != confirmPassword:
                    return render(request, 'changePassword.html',{'changepasswordform':ChangePasswordForm(),'message':'Passwords do not match'})
                salt = bcrypt.gensalt()
                hashedPassword = bcrypt.hashpw(newPassword.encode('utf-8'),salt)
                obj = TeacherCredentials.objects.get(teacher=teacher)
                obj.password = hashedPassword.decode('utf-8')
                obj.salt = salt.decode('utf-8')
                obj.save()
                teacher.password = newPassword
                teacher.save()
                return redirect('teacherHome')
            else:
                return render(request, 'changePassword.html',{'changepasswordform':ChangePasswordForm(),'message':'Old Password is incorrect'})
        else:
            return render(request, 'changePassword.html',{'changepasswordform':ChangePasswordForm(),'message':''})
    elif request.session.has_key('admin'):
        if request.method=="POST":
            oldPassword = request.POST['oldPassword']
            newPassword = request.POST['newPassword']
            confirmPassword = request.POST['confirmPassword']
            admin = AdminCredentials.objects.get(username=request.session['admin'])
            salt = admin.salt
            hashedPassword = bcrypt.hashpw(oldPassword.encode('utf-8'),salt.encode('utf-8'))
            if hashedPassword.decode('utf-8') == admin.hashedPassword:
                if newPassword != confirmPassword:
                    return render(request, 'changePassword.html',{'changepasswordform':ChangePasswordForm(),'message':'Passwords do not match'})
                salt = bcrypt.gensalt()
                hashedPassword = bcrypt.hashpw(newPassword.encode('utf-8'),salt)
                admin.hashedPassword = hashedPassword.decode('utf-8')
                admin.salt = salt.decode('utf-8')
                admin.save()
                return redirect('adminHome')
            else:
                return render(request, 'changePassword.html',{'changepasswordform':ChangePasswordForm(),'message':'Old Password is incorrect'})
        else:
            return render(request, 'changePassword.html',{'changepasswordform':ChangePasswordForm(),'message':''})
    else:
        return redirect('login')


def adminHome(request):
    if request.session.has_key('student'):
        return redirect('studentHome')
    elif request.session.has_key('teacher'):
        return redirect('teacherHome')
    elif request.session.has_key('admin'):
        return render(request,'adminHome.html')
    elif request.method == 'POST':
        username = request.POST['typeUsernameX']
        password = request.POST['typePasswordX']
        try:
            admin = AdminCredentials.objects.get(username=username)
        except AdminCredentials.DoesNotExist:
            return redirect('adminLogin')
        salt = admin.salt
        hashedPassword = bcrypt.hashpw(password.encode('utf-8'),salt.encode('utf-8')).decode('utf-8')
        if hashedPassword != admin.hashedPassword:
            return redirect('adminLogin')
        request.session['admin'] = username
        request.session.modified = True
        request.session.save()
        return render(request,'adminHome.html')
```

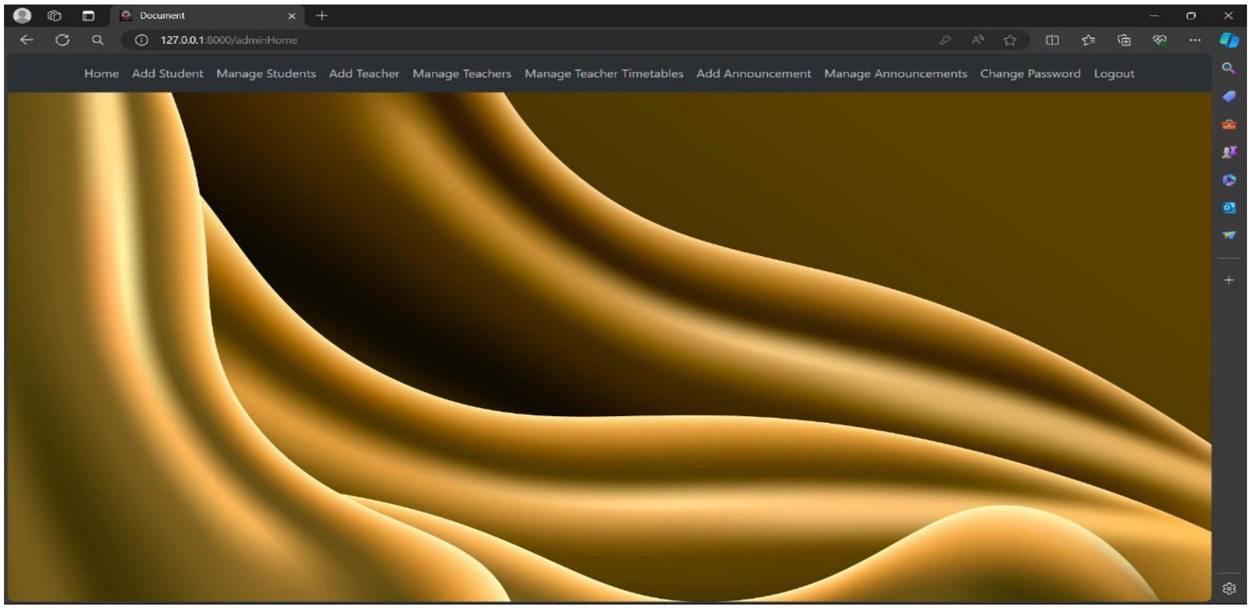
Outputs:

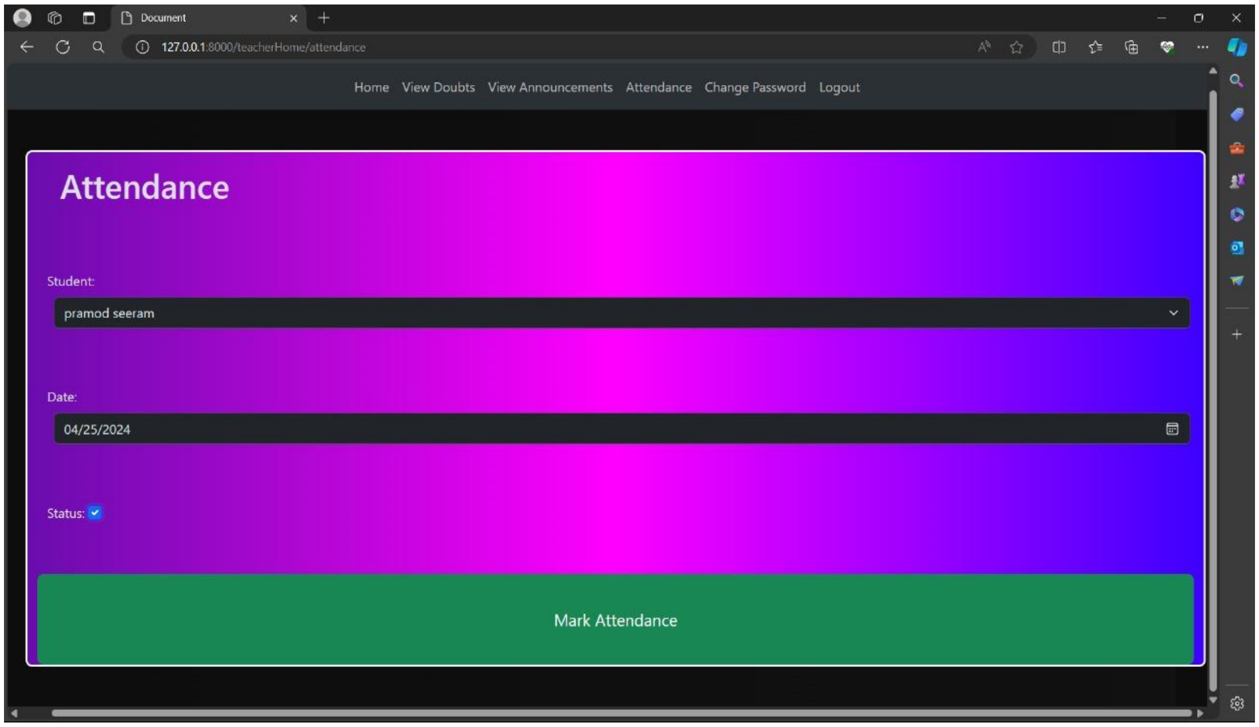

Student
127.0.0.1:8000/studentLogin
STUDENT LOGIN
Please enter your login and password!
Username
Password
Login


Teacher
127.0.0.1:8000/teacherLogin
TEACHER LOGIN
Please enter your login and password!
Username
Password
Login


Admin
127.0.0.1:8000/adminLogin
ADMIN LOGIN
Please enter your login and password!
Username
Password
Login


Login
127.0.0.1:8000
Select Login Type
Student Login
Teacher Login
Admin Login
WELCOME TO COLLEGE MANAGEMENT SYSTEM


Document
127.0.0.1:8000/adminHome
Home Add Student Manage Students Add Teacher Manage Teachers Manage Teacher Timetables Add Announcement Manage Announcements Change Password Logout


Document
127.0.0.1:8000/teacherHome/attendance
Home View Doubts View Announcements Attendance Change Password Logout
Mark Attendance

Attendance

Student: pramod seeram

Date: 04/25/2024

Status: ✓

CONCLUSION

The introduction of an automated Attendance Management System marks a pivotal milestone in the realm of higher education. By addressing the inherent challenges associated with manual attendance tracking, this innovative system offers a transformative solution that enhances the experience of administrators, educators, and student