**Gayatri Vidya Parishad College Of Engineering For Women**

**(Approved by AICTE New Delhi,Affliated to JNTUK Kakinada)**

**[Accredited by National Board of Accreditation(NBA)for CSE,ECE,IT-Valid from 2019-20 to 2021-22] Kommadi,Madhurawada,Visakhapatnam-530048**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Department Of Computer Science and Engineering**

**Hospital Chat Bot For Appointment Booking**

**Submitted By**

**Name**

**1.Ch.Lakshmi Prasanna(18JG1A0517)**

**2.K.Niharika(18JG1A0549)**

**3.Ch.Darshini(18JG1A0516)**

**4.Supritha(18JG1A0556)**

**Under the guidance of**

**Guide Name**

**Mr.K.Lakshman Rao**

**Assistant Professor**

**Signature of guide Signature of coordinator**

**HOSPITAL CHAT BOT**

**ABSTRACT:**

* This is a pure appointment booking chatbot aimed to drive more appointments for hospitals. If you have a call center team handling patient appointments or a generic form on your website, you can easily use this scheduling chatbot template to increase your chances of generating direct bookings.
* A Hospital Appointment Booking Chatbot is a conversational agent that can help patients schedule an appointment with the relevant department without waiting in a queue. It can also answer their appointment related queries in real-time to make sure that patients get the required assistance.
* Like any other business, the modern medical practice needs to be up-to-speed with the preferences of its target clientele. When patients are strapped for time, the last thing they want is to have to visit the doctor’s clinic in person to make an appointment. Even calling the clinic and waiting on the line can be less than ideal sometimes. In any case, neither of these methods matches the simplicity and convenience of typing a quick request into a chat window and getting a confirmed appointment.
* It’s only natural to be met with raised eyebrows while mentioning healthcare and AI chatbots in one sentence. But chatbots are quite the opposite of a threat for the medical industry. They are built to perform basic and repetitive tasks that can streamline the day-to-day functioning of a medical practice. Most of all, they assist with appointment scheduling.
* Think of medical chatbots as virtual assistants for your clinic, hospital, or healthcare center. They’re inexpensive, easy to add to your website, and highly efficient at what they do. To make the picture a little clearer, let’s run through some of the benefits you get when you create a chatbot to make appointments at your clinic.
* There are plenty of Benefits of Hospital Appointment Booking Chatbot, such as:
* Fasten up the process of appointment booking.
* Free up the human agents to address the complex queries of the patients.
* Send the user appointment details directly via Email/SMS to the concerned doctors.

**INTRODUCTION:**

**DJANGO:**

* + Django is a high-level Python web framework that enables rapid development of secure and maintainable websites.
  + Built by experienced developers, Django tasks care of much of the hassle of web development, so much you can focus on writing your app without needing to reinvent the wheel.
  + It is free and open source, has a thriving and active community, great documentation, and many options for free and paid-for support
  + A framework is nothing more than a collection of modules that make development easier.
  + They are grouped together, and allow you to create applications or websites from an existing source, instead of from scratch**.**

**Platform:**

* Visual Studio made by Microsoft for Windows, Linux and Mac OS
* Features include support for debugging, syntax highlighting, intelligent code completion, code refactoring and embedded Git
* Instead of project system, it allows users to open one or more directories, which can then be saved in workspace for future reuse
* This allows it to operate as a language code editor for any language. Visual Studio Code allows users to set the code page in which the active document is saved, the new line character, and the programming language of the active document
* The Visual Studio Express edition products are installed with their own Applds, but the Standard, Professional, and Team Suite products share the same Appld.
* Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

**HTML:**

* HTML stands for Hyper Text Markup Language
* It is used to design web pages using markup  language.
* Hyper Text defines the link between the web pages.
* Markup Language is used to define the text document within tag which defines the structure of web pages.
* **Features of HTML:**
* Easy to learn and easy to use
* Platform independent.
* Images, video and audio can be added to a web page
* Hypertext can be added to text
* Markup language.

**CSS:**

* CSS stands for Cascading Style Sheets.
* CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
* CSS saves a lot of work. It can control the layout of multiple web pages all at once.
* External stylesheets are stored in CSS files.

**PYTHON:**

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

### What can Python do?

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

**JAVASCRIPT:**

**JavaScript** is a text-based programming language **used** both on the client-side and server-side that allows you to make web pages interactive. Where HTML and CSS are languages that give structure and style to web pages, **JavaScript** gives web pages interactive elements that engage a user.

**BOOTSTRAP:**

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

**REQIREMENTS**

**SOFTWARE:**

* CSS
* HTML
* DBSQLITE3
* DJANGO FRAMEWORK
* VS CODE
* PYTHON
* JAVASCRIPT
* jQuery
* BOOTSTRAP

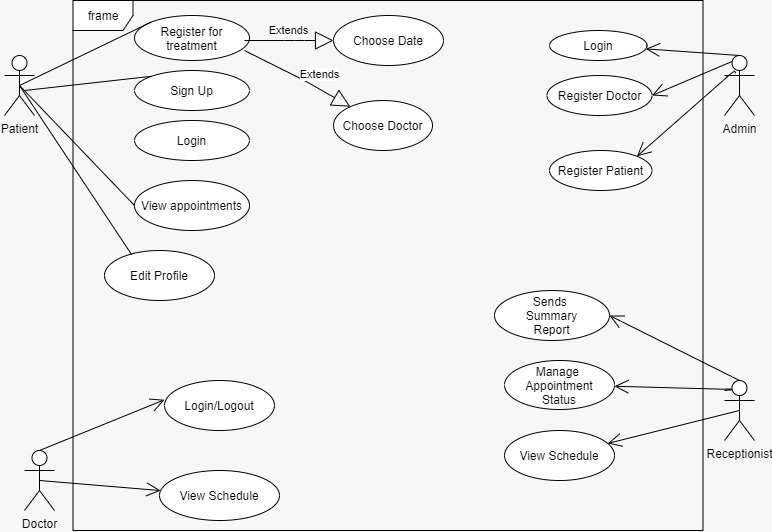
**Hardware Requirements**

● Core i5 processor

● 4GB Ram

● 20GB of hard disk space in terminal machines

**Design**



**Code:**

**App name**: Hospital chat bot

**Project name**: Hospital Management website with chatbot

**Preparing system using CLI:**

Installing Virtual environment in the system:

pip install virtualenvwrapper-win

Creating the virtual environment: mkvirtualenv <env\_name>

Activating the virtual environment: workon <env\_name>

Installing DJANGO in the virtual Environment:

pip install django

### Some of the packages that are required:

pip install django-widget-tweaks==1.4.8

pip install xhtml2pdf

pip install sqlparse==0.3.1

**views.py**

from django.shortcuts import render,redirect,reverse

from . import forms,models

from django.db.models import Sum

from django.contrib.auth.models import Group

from django.http import HttpResponseRedirect

from django.core.mail import send\_mail

from django.contrib.auth.decorators import login\_required,user\_passes\_test

from datetime import datetime,timedelta,date

from django.conf import settings

# Create your views here.

def home\_view(request):

if request.user.is\_authenticated:

return HttpResponseRedirect('afterlogin')

return render(request,'hospital/index.html')

#for showing signup/login button

def adminclick\_view(request):

if request.user.is\_authenticated:

return HttpResponseRedirect('afterlogin')

return render(request,'hospital/adminclick.html')

#for showing signup/login button for doctor

def doctorclick\_view(request):

if request.user.is\_authenticated:

return HttpResponseRedirect('afterlogin')

return render(request,'hospital/doctorclick.html')

#for showing signup/login button for patient

def patientclick\_view(request):

if request.user.is\_authenticated:

return HttpResponseRedirect('afterlogin')

return render(request,'hospital/patientclick.html')

def admin\_signup\_view(request):

form=forms.AdminSigupForm()

if request.method=='POST':

form=forms.AdminSigupForm(request.POST)

if form.is\_valid():

user=form.save()

user.set\_password(user.password)

user.save()

my\_admin\_group = Group.objects.get\_or\_create(name='ADMIN')

my\_admin\_group[0].user\_set.add(user)

return HttpResponseRedirect('adminlogin')

return render(request,'hospital/adminsignup.html',{'form':form})

def doctor\_signup\_view(request):

userForm=forms.DoctorUserForm()

doctorForm=forms.DoctorForm()

mydict={'userForm':userForm,'doctorForm':doctorForm}

if request.method=='POST':

userForm=forms.DoctorUserForm(request.POST)

doctorForm=forms.DoctorForm(request.POST,request.FILES)

if userForm.is\_valid() and doctorForm.is\_valid():

user=userForm.save()

user.set\_password(user.password)

user.save()

doctor=doctorForm.save(commit=False)

doctor.user=user

doctor=doctor.save()

my\_doctor\_group = Group.objects.get\_or\_create(name='DOCTOR')

my\_doctor\_group[0].user\_set.add(user)

return HttpResponseRedirect('doctorlogin')

return render(request,'hospital/doctorsignup.html',context=mydict)

def patient\_signup\_view(request):

userForm=forms.PatientUserForm()

patientForm=forms.PatientForm()

mydict={'userForm':userForm,'patientForm':patientForm}

if request.method=='POST':

userForm=forms.PatientUserForm(request.POST)

patientForm=forms.PatientForm(request.POST,request.FILES)

if userForm.is\_valid() and patientForm.is\_valid():

user=userForm.save()

user.set\_password(user.password)

user.save()

patient=patientForm.save(commit=False)

patient.user=user

patient.assignedDoctorId=request.POST.get('assignedDoctorId')

patient=patient.save()

my\_patient\_group = Group.objects.get\_or\_create(name='PATIENT')

my\_patient\_group[0].user\_set.add(user)

return HttpResponseRedirect('patientlogin')

return render(request,'hospital/patientsignup.html',context=mydict)

#-----------for checking user is doctor , patient or admin

def is\_admin(user):

return user.groups.filter(name='ADMIN').exists()

def is\_doctor(user):

return user.groups.filter(name='DOCTOR').exists()

def is\_patient(user):

return user.groups.filter(name='PATIENT').exists()

def afterlogin\_view(request):

if is\_admin(request.user):

return redirect('admin-dashboard')

elif is\_doctor(request.user):

accountapproval=models.Doctor.objects.all().filter(user\_id=request.user.id,status=True)

if accountapproval:

return redirect('doctor-dashboard')

else:

return render(request,'hospital/doctor\_wait\_for\_approval.html')

elif is\_patient(request.user):

accountapproval=models.Patient.objects.all().filter(user\_id=request.user.id,status=True)

if accountapproval:

return redirect('patient-dashboard')

else:

return render(request,'hospital/patient\_wait\_for\_approval.html')

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_dashboard\_view(request):

#for both table in admin dashboard

doctors=models.Doctor.objects.all().order\_by('-id')

patients=models.Patient.objects.all().order\_by('-id')

#for three cards

doctorcount=models.Doctor.objects.all().filter(status=True).count()

pendingdoctorcount=models.Doctor.objects.all().filter(status=False).count() patientcount=models.Patient.objects.all().filter(status=True).count()

pendingpatientcount=models.Patient.objects.all().filter(status=False).count()

appointmentcount=models.Appointment.objects.all().filter(status=True).count()

pendingappointmentcount=models.Appointment.objects.all().filter(status=False).count() mydict={

'doctors':doctors,

'patients':patients,

'doctorcount':doctorcount,

'pendingdoctorcount':pendingdoctorcount,

'patientcount':patientcount,

'pendingpatientcount':pendingpatientcount,

'appointmentcount':appointmentcount,

'pendingappointmentcount':pendingappointmentcount,

}

return render(request,'hospital/admin\_dashboard.html',context=mydict)

# this view for sidebar click on admin page

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_doctor\_view(request):

return render(request,'hospital/admin\_doctor.html')

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_view\_doctor\_view(request):

doctors=models.Doctor.objects.all().filter(status=True)

return render(request,'hospital/admin\_view\_doctor.html',{'doctors':doctors})

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def delete\_doctor\_from\_hospital\_view(request,pk):

doctor=models.Doctor.objects.get(id=pk)

user=models.User.objects.get(id=doctor.user\_id)

user.delete()

doctor.delete()

return redirect('admin-view-doctor')

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def update\_doctor\_view(request,pk):

doctor=models.Doctor.objects.get(id=pk)

user=models.User.objects.get(id=doctor.user\_id)

userForm=forms.DoctorUserForm(instance=user)

doctorForm=forms.DoctorForm(request.FILES,instance=doctor)

mydict={'userForm':userForm,'doctorForm':doctorForm}

if request.method=='POST':

userForm=forms.DoctorUserForm(request.POST,instance=user)

doctorForm=forms.DoctorForm(request.POST,request.FILES,instance=doctor)

if userForm.is\_valid() and doctorForm.is\_valid():

user=userForm.save()

user.set\_password(user.password)

user.save()

doctor=doctorForm.save(commit=False)

doctor.status=True

doctor.save()

return redirect('admin-view-doctor')

return render(request,'hospital/admin\_update\_doctor.html',context=mydict)

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_add\_doctor\_view(request):

userForm=forms.DoctorUserForm()

doctorForm=forms.DoctorForm()

mydict={'userForm':userForm,'doctorForm':doctorForm}

if request.method=='POST':

userForm=forms.DoctorUserForm(request.POST)

doctorForm=forms.DoctorForm(request.POST, request.FILES)

if userForm.is\_valid() and doctorForm.is\_valid():

user=userForm.save()

user.set\_password(user.password)

user.save()

doctor=doctorForm.save(commit=False)

doctor.user=user

doctor.status=True

doctor.save()

my\_doctor\_group = Group.objects.get\_or\_create(name='DOCTOR')

my\_doctor\_group[0].user\_set.add(user)

return HttpResponseRedirect('admin-view-doctor')

return render(request,'hospital/admin\_add\_doctor.html',context=mydict)

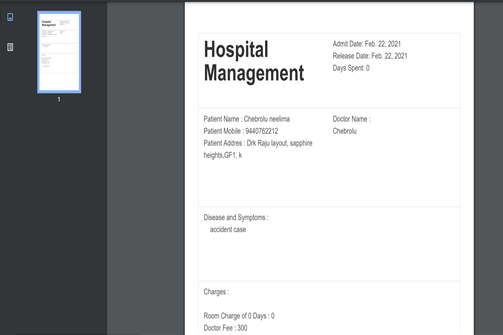
@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_approve\_doctor\_view(request):

#those whose approval are needed

doctors=models.Doctor.objects.all().filter(status=False)

return render(request,'hospital/admin\_approve\_doctor.html',{'doctors':doctors})

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def approve\_doctor\_view(request,pk):

doctor=models.Doctor.objects.get(id=pk)

doctor.status=True

doctor.save()

return redirect(reverse('admin-approve-doctor'))

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def reject\_doctor\_view(request,pk):

doctor=models.Doctor.objects.get(id=pk)

user=models.User.objects.get(id=doctor.user\_id)

user.delete()

doctor.delete()

return redirect('admin-approve-doctor')

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_view\_doctor\_specialisation\_view(request):

doctors=models.Doctor.objects.all().filter(status=True)

return render(request,'hospital/admin\_view\_doctor\_specialisation.html',{'doctors':doctors})

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_patient\_view(request):

return render(request,'hospital/admin\_patient.html')

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_view\_patient\_view(request):

patients=models.Patient.objects.all().filter(status=True)

return render(request,'hospital/admin\_view\_patient.html',{'patients':patients})

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def delete\_patient\_from\_hospital\_view(request,pk):

patient=models.Patient.objects.get(id=pk)

user=models.User.objects.get(id=patient.user\_id)

user.delete()

patient.delete()

return redirect('admin-view-patient')

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def update\_patient\_view(request,pk):

patient=models.Patient.objects.get(id=pk)

user=models.User.objects.get(id=patient.user\_id)

userForm=forms.PatientUserForm(instance=user)

patientForm=forms.PatientForm(request.FILES,instance=patient)

mydict={'userForm':userForm,'patientForm':patientForm}

if request.method=='POST':

userForm=forms.PatientUserForm(request.POST,instance=user)

patientForm=forms.PatientForm(request.POST,request.FILES,instance=patient)

if userForm.is\_valid() and patientForm.is\_valid():

user=userForm.save()

user.set\_password(user.password)

user.save()

patient=patientForm.save(commit=False)

patient.status=True

patient.assignedDoctorId=request.POST.get('assignedDoctorId')

patient.save()

return redirect('admin-view-patient')

return render(request,'hospital/admin\_update\_patient.html',context=mydict)

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_add\_patient\_view(request):

userForm=forms.PatientUserForm()

patientForm=forms.PatientForm()

mydict={'userForm':userForm,'patientForm':patientForm}

if request.method=='POST':

userForm=forms.PatientUserForm(request.POST)

patientForm=forms.PatientForm(request.POST,request.FILES)

if userForm.is\_valid() and patientForm.is\_valid():

user=userForm.save()

user.set\_password(user.password)

user.save()

patient=patientForm.save(commit=False)

patient.user=user

patient.status=True

patient.assignedDoctorId=request.POST.get('assignedDoctorId')

patient.save()

my\_patient\_group = Group.objects.get\_or\_create(name='PATIENT')

my\_patient\_group[0].user\_set.add(user)

return HttpResponseRedirect('admin-view-patient')

return render(request,'hospital/admin\_add\_patient.html',context=mydict)

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_approve\_patient\_view(request):

#those whose approval are needed

patients=models.Patient.objects.all().filter(status=False)

return render(request,'hospital/admin\_approve\_patient.html',{'patients':patients})

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def approve\_patient\_view(request,pk):

patient=models.Patient.objects.get(id=pk)

patient.status=True

patient.save()

return redirect(reverse('admin-approve-patient'))

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def reject\_patient\_view(request,pk):

patient=models.Patient.objects.get(id=pk)

user=models.User.objects.get(id=patient.user\_id)

user.delete()

patient.delete()

return redirect('admin-approve-patient')

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_discharge\_patient\_view(request):

patients=models.Patient.objects.all().filter(status=True)

return render(request,'hospital/admin\_discharge\_patient.html',{'patients':patients})

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def discharge\_patient\_view(request,pk):

patient=models.Patient.objects.get(id=pk)

days=(date.today()-patient.admitDate) #2 days, 0:00:00

assignedDoctor=models.User.objects.all().filter(id=patient.assignedDoctorId)

d=days.days # only how many day that is 2

patientDict={

'patientId':pk,

'name':patient.get\_name,

'mobile':patient.mobile,

'address':patient.address,

'symptoms':patient.symptoms,

'admitDate':patient.admitDate,

'todayDate':date.today(),

'day':d,

'assignedDoctorName':assignedDoctor[0].first\_name,

}

if request.method == 'POST':

feeDict ={

'roomCharge':int(request.POST['roomCharge'])\*int(d),

'doctorFee':request.POST['doctorFee'],

'medicineCost' : request.POST['medicineCost'],

'OtherCharge' : request.POST['OtherCharge'],

'total':(int(request.POST['roomCharge'])\*int(d))+int(request.POST['doctorFee'])+int(request.POST['medicineCost'])+int(request.POST['OtherCharge'])

}

patientDict.update(feeDict)

#for updating to database patientDischargeDetails (pDD)

pDD=models.PatientDischargeDetails()

pDD.patientId=pk

pDD.patientName=patient.get\_name

pDD.assignedDoctorName=assignedDoctor[0].first\_name

pDD.address=patient.address

pDD.mobile=patient.mobile

pDD.symptoms=patient.symptoms

pDD.admitDate=patient.admitDate

pDD.releaseDate=date.today()

pDD.daySpent=int(d)

pDD.medicineCost=int(request.POST['medicineCost'])

pDD.roomCharge=int(request.POST['roomCharge'])\*int(d)

pDD.doctorFee=int(request.POST['doctorFee'])

pDD.OtherCharge=int(request.POST['OtherCharge'])

pDD.total=(int(request.POST['roomCharge'])\*int(d))+int(request.POST['doctorFee'])+int(request.POST['medicineCost'])+int(request.POST['OtherCharge'])

pDD.save()

return render(request,'hospital/patient\_final\_bill.html',context=patientDict)

return render(request,'hospital/patient\_generate\_bill.html',context=patientDict

import io

from xhtml2pdf import pisa

from django.template.loader import get\_template

from django.template import Context

from django.http import HttpResponse

def render\_to\_pdf(template\_src, context\_dict):

template = get\_template(template\_src)

html = template.render(context\_dict)

result = io.BytesIO()

pdf = pisa.pisaDocument(io.BytesIO(html.encode("ISO-8859-1")), result)

if not pdf.err:

return HttpResponse(result.getvalue(), content\_type='application/pdf')

return

def download\_pdf\_view(request,pk):

dischargeDetails=models.PatientDischargeDetails.objects.all().filter(patientId=pk).order\_by('-id')[:1]

dict={

'patientName':dischargeDetails[0].patientName,

'assignedDoctorName':dischargeDetails[0].assignedDoctorName,

'address':dischargeDetails[0].address,

'mobile':dischargeDetails[0].mobile,

'symptoms':dischargeDetails[0].symptoms,

'admitDate':dischargeDetails[0].admitDate,

'releaseDate':dischargeDetails[0].releaseDate,

'daySpent':dischargeDetails[0].daySpent,

'medicineCost':dischargeDetails[0].medicineCost,

'roomCharge':dischargeDetails[0].roomCharge,

'doctorFee':dischargeDetails[0].doctorFee,

'OtherCharge':dischargeDetails[0].OtherCharge,

'total':dischargeDetails[0].total,

}

return render\_to\_pdf('hospital/download\_bill.html',dict)

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_appointment\_view(request):

return render(request,'hospital/admin\_appointment.html')

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_view\_appointment\_view(request):

appointments=models.Appointment.objects.all().filter(status=True)

return render(request,'hospital/admin\_view\_appointment.html',{'appointments':appointments})

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_add\_appointment\_view(request):

appointmentForm=forms.AppointmentForm()

mydict={'appointmentForm':appointmentForm,}

if request.method=='POST':

appointmentForm=forms.AppointmentForm(request.POST)

if appointmentForm.is\_valid():

appointment=appointmentForm.save(commit=False)

appointment.doctorId=request.POST.get('doctorId')

appointment.patientId=request.POST.get('patientId')

appointment.doctorName=models.User.objects.get(id=request.POST.get('doctorId')).first\_name

appointment.patientName=models.User.objects.get(id=request.POST.get('patientId')).first\_name

appointment.status=True

appointment.save()

return HttpResponseRedirect('admin-view-appointment')

return render(request,'hospital/admin\_add\_appointment.html',context=mydict)

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def admin\_approve\_appointment\_view(request):

#those whose approval are needed

appointments=models.Appointment.objects.all().filter(status=False)

return render(request,'hospital/admin\_approve\_appointment.html',{'appointments':appointments})

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def approve\_appointment\_view(request,pk):

appointment=models.Appointment.objects.get(id=pk)

appointment.status=True

appointment.save()

return redirect(reverse('admin-approve-appointment'))

@login\_required(login\_url='adminlogin')

@user\_passes\_test(is\_admin)

def reject\_appointment\_view(request,pk):

appointment=models.Appointment.objects.get(id=pk)

appointment.delete()

return redirect('admin-approve-appointment')

@login\_required(login\_url='doctorlogin')

@user\_passes\_test(is\_doctor)

def doctor\_dashboard\_view(request):

#for three cards

patientcount=models.Patient.objects.all().filter(status=True,assignedDoctorId=request.user.id).count()

appointmentcount=models.Appointment.objects.all().filter(status=True,doctorId=request.user.id).count()

patientdischarged=models.PatientDischargeDetails.objects.all().distinct().filter(assignedDoctorName=request.user.first\_name).count()

appointments=models.Appointment.objects.all().filter(status=True,doctorId=request.user.id).order\_by('-id')

patientid=[]

for a in appointments:

patientid.append(a.patientId)

patients=models.Patient.objects.all().filter(status=True,user\_id\_\_in=patientid).order\_by('-id')

appointments=zip(appointments,patients)

mydict={

'patientcount':patientcount,

'appointmentcount':appointmentcount,

'patientdischarged':patientdischarged,

'appointments':appointments,

'doctor':models.Doctor.objects.get(user\_id=request.user.id), #for profile picture of doctor in sidebar

}

return render(request,'hospital/doctor\_dashboard.html',context=mydict)

@login\_required(login\_url='doctorlogin')

@user\_passes\_test(is\_doctor)

def doctor\_patient\_view(request):

mydict={

'doctor':models.Doctor.objects.get(user\_id=request.user.id), #for profile picture of doctor in sidebar

}

return render(request,'hospital/doctor\_patient.html',context=mydict)

@login\_required(login\_url='doctorlogin')

@user\_passes\_test(is\_doctor)

def doctor\_view\_patient\_view(request):

patients=models.Patient.objects.all().filter(status=True,assignedDoctorId=request.user.id)

doctor=models.Doctor.objects.get(user\_id=request.user.id) #for profile picture of doctor in sidebar

return render(request,'hospital/doctor\_view\_patient.html',{'patients':patients,'doctor':doctor})

@login\_required(login\_url='doctorlogin')

@user\_passes\_test(is\_doctor)

def doctor\_view\_discharge\_patient\_view(request):

dischargedpatients=models.PatientDischargeDetails.objects.all().distinct().filter(assignedDoctorName=request.user.first\_name)

doctor=models.Doctor.objects.get(user\_id=request.user.id) #for profile picture of doctor in sidebar

return render(request,'hospital/doctor\_view\_discharge\_patient.html',{'dischargedpatients':dischargedpatients,'doctor':doctor})

@login\_required(login\_url='doctorlogin')

@user\_passes\_test(is\_doctor)

def doctor\_appointment\_view(request):

doctor=models.Doctor.objects.get(user\_id=request.user.id) #for profile picture of doctor in sidebar

return render(request,'hospital/doctor\_appointment.html',{'doctor':doctor})

@login\_required(login\_url='doctorlogin')

@user\_passes\_test(is\_doctor)

def doctor\_view\_appointment\_view(request):

doctor=models.Doctor.objects.get(user\_id=request.user.id) #for profile picture of doctor in sidebar

appointments=models.Appointment.objects.all().filter(status=True,doctorId=request.user.id)

patientid=[]

for a in appointments:

patientid.append(a.patientId)

patients=models.Patient.objects.all().filter(status=True,user\_id\_\_in=patientid)

appointments=zip(appointments,patients)

return render(request,'hospital/doctor\_view\_appointment.html',{'appointments':appointments,'doctor':doctor})

@login\_required(login\_url='doctorlogin')

@user\_passes\_test(is\_doctor)

def doctor\_delete\_appointment\_view(request):

doctor=models.Doctor.objects.get(user\_id=request.user.id) #for profile picture of doctor in sidebar

appointments=models.Appointment.objects.all().filter(status=True,doctorId=request.user.id)

patientid=[]

for a in appointments:

patientid.append(a.patientId)

patients=models.Patient.objects.all().filter(status=True,user\_id\_\_in=patientid)

appointments=zip(appointments,patients)

return render(request,'hospital/doctor\_delete\_appointment.html',{'appointments':appointments,'doctor':doctor})

@login\_required(login\_url='doctorlogin')

@user\_passes\_test(is\_doctor)

def delete\_appointment\_view(request,pk):

appointment=models.Appointment.objects.get(id=pk)

appointment.delete()

doctor=models.Doctor.objects.get(user\_id=request.user.id) #for profile picture of doctor in sidebar

appointments=models.Appointment.objects.all().filter(status=True,doctorId=request.user.id)

patientid=[]

for a in appointments:

patientid.append(a.patientId)

patients=models.Patient.objects.all().filter(status=True,user\_id\_\_in=patientid)

appointments=zip(appointments,patients)

return render(request,'hospital/doctor\_delete\_appointment.html',{'appointments':appointments,'doctor':doctor})

@login\_required(login\_url='patientlogin')

@user\_passes\_test(is\_patient)

def patient\_dashboard\_view(request):

patient=models.Patient.objects.get(user\_id=request.user.id)

doctor=models.Doctor.objects.get(user\_id=patient.assignedDoctorId)

mydict={

'patient':patient,

'doctorName':doctor.get\_name,

'doctorMobile':doctor.mobile,

'doctorAddress':doctor.address,

'symptoms':patient.symptoms,

'doctorDepartment':doctor.department,

'admitDate':patient.admitDate,

}

return render(request,'hospital/patient\_dashboard.html',context=mydict)

@login\_required(login\_url='patientlogin')

@user\_passes\_test(is\_patient)

def patient\_appointment\_view(request):

patient=models.Patient.objects.get(user\_id=request.user.id) #for profile picture of patient in sidebar

return render(request,'hospital/patient\_appointment.html',{'patient':patient})

@login\_required(login\_url='patientlogin')

@user\_passes\_test(is\_patient)

def patient\_book\_appointment\_view(request):

appointmentForm=forms.PatientAppointmentForm()

patient=models.Patient.objects.get(user\_id=request.user.id) #for profile picture of patient in sidebar

mydict={'appointmentForm':appointmentForm,'patient':patient}

if request.method=='POST':

appointmentForm=forms.PatientAppointmentForm(request.POST)

if appointmentForm.is\_valid():

appointment=appointmentForm.save(commit=False)

appointment.doctorId=request.POST.get('doctorId')

appointment.patientId=request.user.id #----user can choose any patient but only their info will be stored

appointment.doctorName=models.User.objects.get(id=request.POST.get('doctorId')).first\_name

appointment.patientName=request.user.first\_name #----user can choose any patient but only their info will be stored

appointment.status=False

appointment.save()

return HttpResponseRedirect('patient-view-appointment')

return render(request,'hospital/patient\_book\_appointment.html',context=mydict)

@login\_required(login\_url='patientlogin')

@user\_passes\_test(is\_patient)

def patient\_view\_appointment\_view(request):

patient=models.Patient.objects.get(user\_id=request.user.id) #for profile picture of patient in sidebar

appointments=models.Appointment.objects.all().filter(patientId=request.user.id)

return render(request,'hospital/patient\_view\_appointment.html',{'appointments':appointments,'patient':patient})

@login\_required(login\_url='patientlogin')

@user\_passes\_test(is\_patient)

def patient\_discharge\_view(request):

patient=models.Patient.objects.get(user\_id=request.user.id) #for profile picture of patient in sidebar

dischargeDetails=models.PatientDischargeDetails.objects.all().filter(patientId=patient.id).order\_by('-id')[:1]

patientDict=None

if dischargeDetails:

patientDict ={

'is\_discharged':True,

'patient':patient,

'patientId':patient.id,

'patientName':patient.get\_name,

'assignedDoctorName':dischargeDetails[0].assignedDoctorName,

'address':patient.address,

'mobile':patient.mobile,

'symptoms':patient.symptoms,

'admitDate':patient.admitDate,

'releaseDate':dischargeDetails[0].releaseDate,

'daySpent':dischargeDetails[0].daySpent,

'medicineCost':dischargeDetails[0].medicineCost,

'roomCharge':dischargeDetails[0].roomCharge,

'doctorFee':dischargeDetails[0].doctorFee,

'OtherCharge':dischargeDetails[0].OtherCharge,

'total':dischargeDetails[0].total,

}

print(patientDict)

else:

patientDict={

'is\_discharged':False,

'patient':patient,

'patientId':request.user.id,

}

return render(request,'hospital/patient\_discharge.html',context=patientDict)

def aboutus\_view(request):

return render(request,'hospital/aboutus.html')

def contactus\_view(request):

sub = forms.ContactusForm()

if request.method == 'POST':

sub = forms.ContactusForm(request.POST)

if sub.is\_valid():

email = sub.cleaned\_data['Email']

name=sub.cleaned\_data['Name']

message = sub.cleaned\_data['Message']

send\_mail(str(name)+' || '+str(email),message,settings.EMAIL\_HOST\_USER, settings.EMAIL\_RECEIVING\_USER, fail\_silently = False)

return render(request, 'hospital/contactussuccess.html')

return render(request, 'hospital/contactus.html', {'form':sub})

**models.py**

from django.db import models

from django.contrib.auth.models import User

departments=[('Cardiologist','Cardiologist'),

('Dermatologists','Dermatologists'),

('Emergency Medicine Specialists','Emergency Medicine Specialists'),

('Allergists/Immunologists','Allergists/Immunologists'),

('Anesthesiologists','Anesthesiologists'),

('Colon and Rectal Surgeons','Colon and Rectal Surgeons')

]

class Doctor(models.Model):

user=models.OneToOneField(User,on\_delete=models.CASCADE)

profile\_pic= models.ImageField(upload\_to='profile\_pic/DoctorProfilePic/',null=True,blank=True)

address = models.CharField(max\_length=40)

mobile = models.CharField(max\_length=20,null=True)

department= models.CharField(max\_length=50,choices=departments,default='Cardiologist')

status=models.BooleanField(default=False)

@property

def get\_name(self):

return self.user.first\_name+" "+self.user.last\_name

@property

def get\_id(self):

return self.user.id

def \_\_str\_\_(self):

return "{} ({})".format(self.user.first\_name,self.department)

class Patient(models.Model):

user=models.OneToOneField(User,on\_delete=models.CASCADE)

profile\_pic= models.ImageField(upload\_to='profile\_pic/PatientProfilePic/',null=True,blank=True)

address = models.CharField(max\_length=40)

mobile = models.CharField(max\_length=20,null=False)

symptoms = models.CharField(max\_length=100,null=False)

assignedDoctorId = models.PositiveIntegerField(null=True)

admitDate=models.DateField(auto\_now=True)

status=models.BooleanField(default=False)

@property

def get\_name(self):

return self.user.first\_name+" "+self.user.last\_name

@property

def get\_id(self):

return self.user.id

def \_\_str\_\_(self):

return self.user.first\_name+" ("+self.symptoms+")"

class Appointment(models.Model):

patientId=models.PositiveIntegerField(null=True)

doctorId=models.PositiveIntegerField(null=True)

patientName=models.CharField(max\_length=40,null=True)

doctorName=models.CharField(max\_length=40,null=True)

appointmentDate=models.DateField(auto\_now=True)

description=models.TextField(max\_length=500)

status=models.BooleanField(default=False)

class PatientDischargeDetails(models.Model):

patientId=models.PositiveIntegerField(null=True)

patientName=models.CharField(max\_length=40)

assignedDoctorName=models.CharField(max\_length=40)

address = models.CharField(max\_length=40)

mobile = models.CharField(max\_length=20,null=True)

symptoms = models.CharField(max\_length=100,null=True)

admitDate=models.DateField(null=False)

releaseDate=models.DateField(null=False)

daySpent=models.PositiveIntegerField(null=False)

roomCharge=models.PositiveIntegerField(null=False)

medicineCost=models.PositiveIntegerField(null=False)

doctorFee=models.PositiveIntegerField(null=False)

OtherCharge=models.PositiveIntegerField(null=False)

total=models.PositiveIntegerField(null=False)

**forms.py**

from django import forms

from django.contrib.auth.models import User

from . import models

#for admin signup

class AdminSigupForm(forms.ModelForm):

class Meta:

model=User

fields=['first\_name','last\_name','username','password']

widgets = {

'password': forms.PasswordInput()

}

#for student related form

class DoctorUserForm(forms.ModelForm):

class Meta:

model=User

fields=['first\_name','last\_name','username','password']

widgets = {

'password': forms.PasswordInput()

}

class DoctorForm(forms.ModelForm):

class Meta:

model=models.Doctor

fields=['address','mobile','department','status','profile\_pic']

#for teacher related form

class PatientUserForm(forms.ModelForm):

class Meta:

model=User

fields=['first\_name','last\_name','username','password']

widgets = {

'password': forms.PasswordInput()

}

class PatientForm(forms.ModelForm):

#this is the extrafield for linking patient and their assigend doctor

#this will show dropdown \_\_str\_\_ method doctor model is shown on html so override it

#to\_field\_name this will fetch corresponding value user\_id present in Doctor model and return it

assignedDoctorId=forms.ModelChoiceField(queryset=models.Doctor.objects.all().filter(status=True),empty\_label="Name and Department", to\_field\_name="user\_id")

class Meta:

model=models.Patient

fields=['address','mobile','status','symptoms','profile\_pic']

class AppointmentForm(forms.ModelForm):

doctorId=forms.ModelChoiceField(queryset=models.Doctor.objects.all().filter(status=True),empty\_label="Doctor Name and Department", to\_field\_name="user\_id")

patientId=forms.ModelChoiceField(queryset=models.Patient.objects.all().filter(status=True),empty\_label="Patient Name and Symptoms", to\_field\_name="user\_id")

class Meta:

model=models.Appointment

fields=['description','status']

class PatientAppointmentForm(forms.ModelForm):

doctorId=forms.ModelChoiceField(queryset=models.Doctor.objects.all().filter(status=True),empty\_label="Doctor Name and Department", to\_field\_name="user\_id")

class Meta:

model=models.Appointment

fields=['description','status']

#for contact us page

class ContactusForm(forms.Form):

Name = forms.CharField(max\_length=30)

Email = forms.EmailField()

Message = forms.CharField(max\_length=500,widget=forms.Textarea(attrs={'rows': 3, 'cols': 30}))

**admin**.**py**

from django.contrib import admin

from .models import Doctor,Patient,Appointment,PatientDischargeDetails

# Register your models here.

class DoctorAdmin(admin.ModelAdmin):

pass

admin.site.register(Doctor, DoctorAdmin)

class PatientAdmin(admin.ModelAdmin):

pass

admin.site.register(Patient, PatientAdmin)

class AppointmentAdmin(admin.ModelAdmin):

pass

admin.site.register(Appointment, AppointmentAdmin)

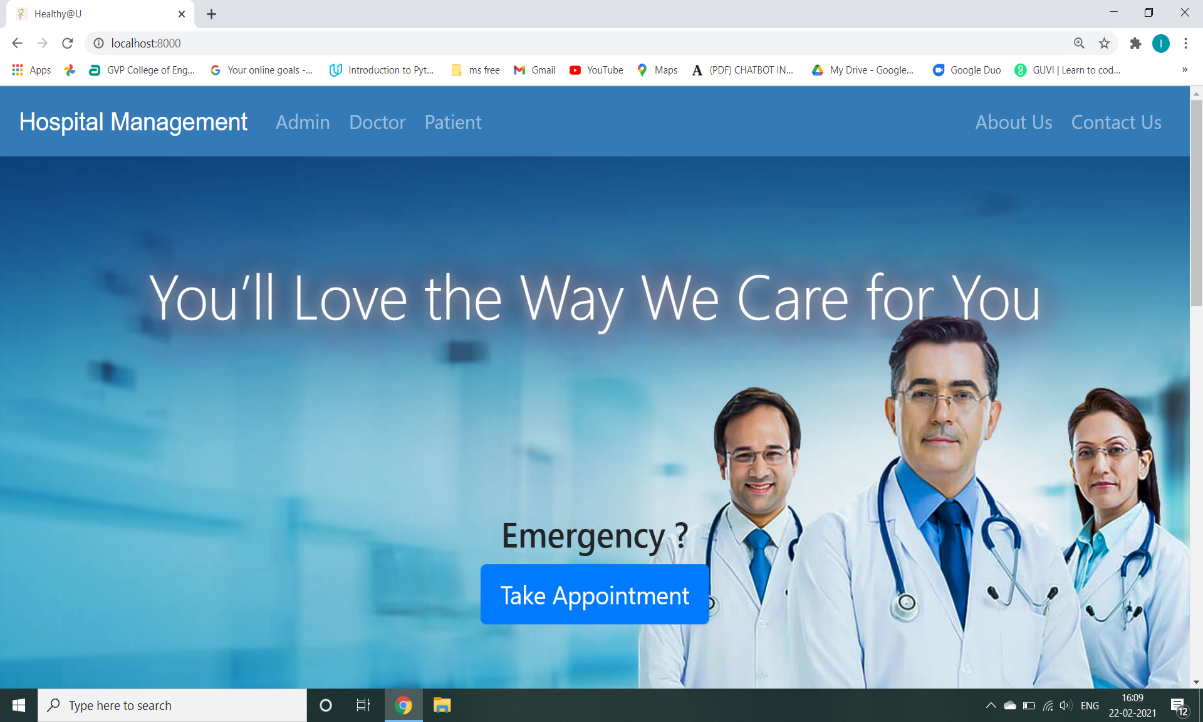
class PatientDischargeDetailsAdmin(admin.ModelAdmin):

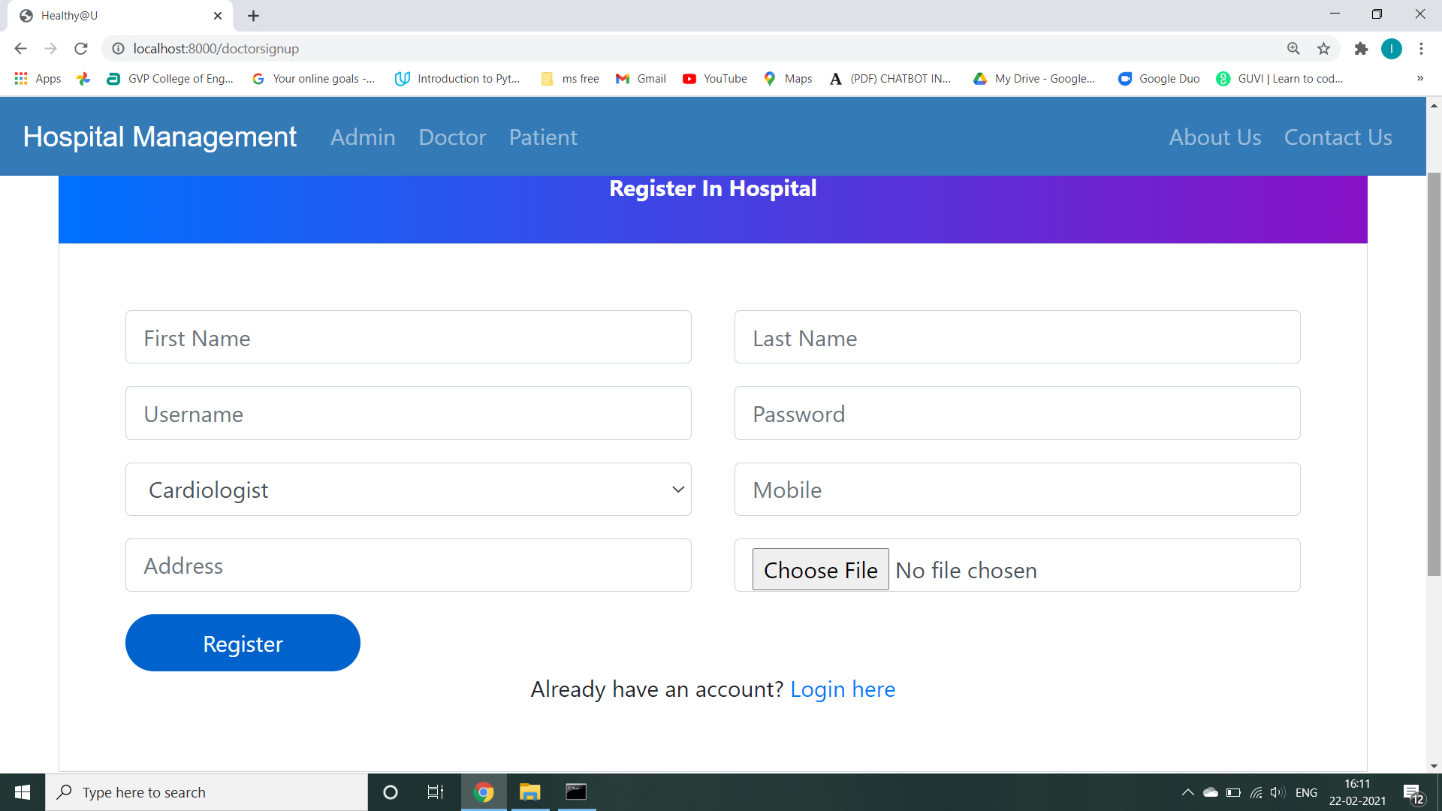
pass

admin.site.register(PatientDischargeDetails, PatientDischargeDetailsAdmin)

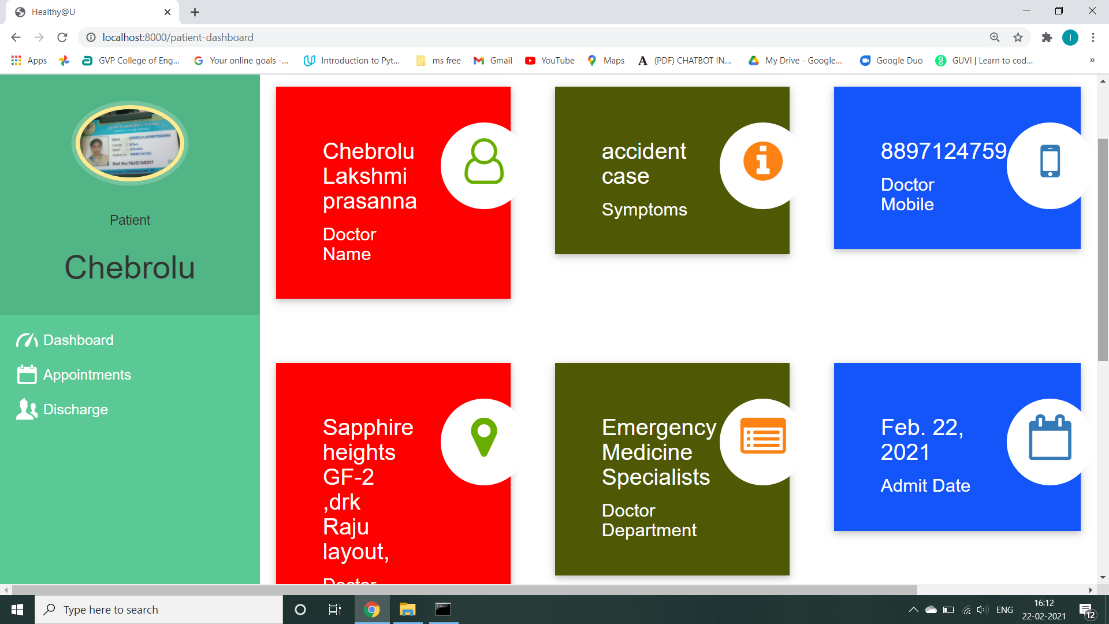
**OUTPUT:**

**HomePage:**

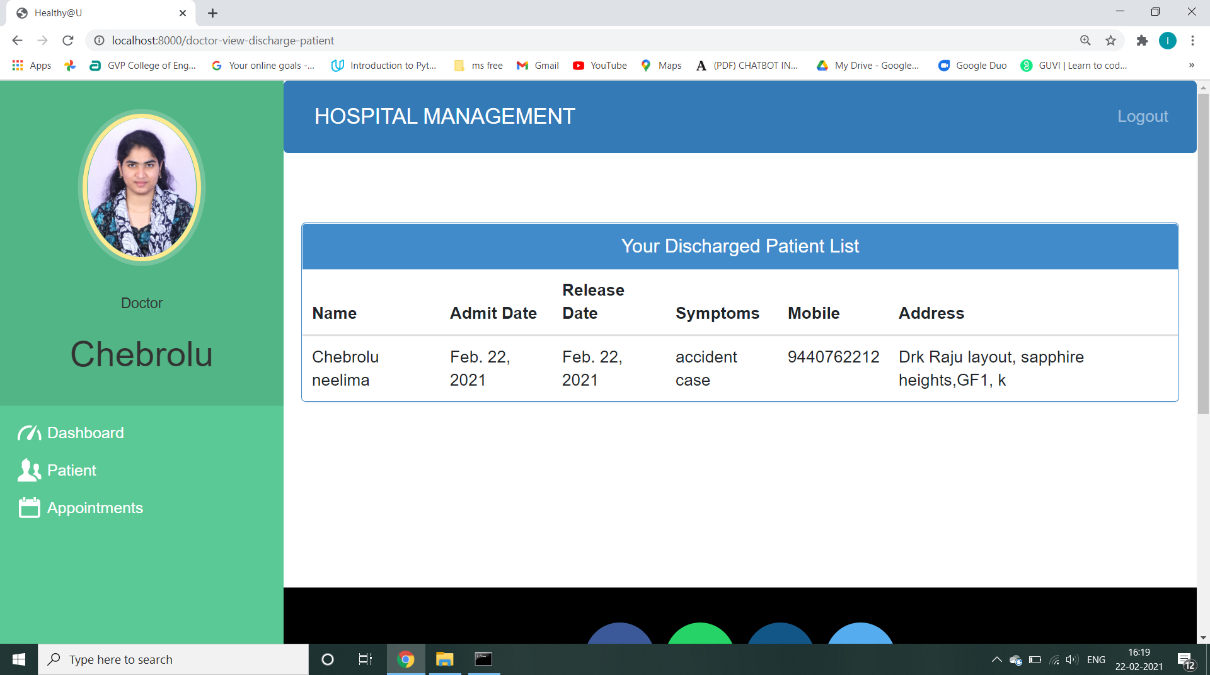


Login page:

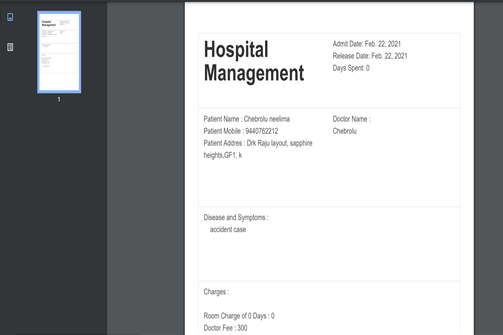
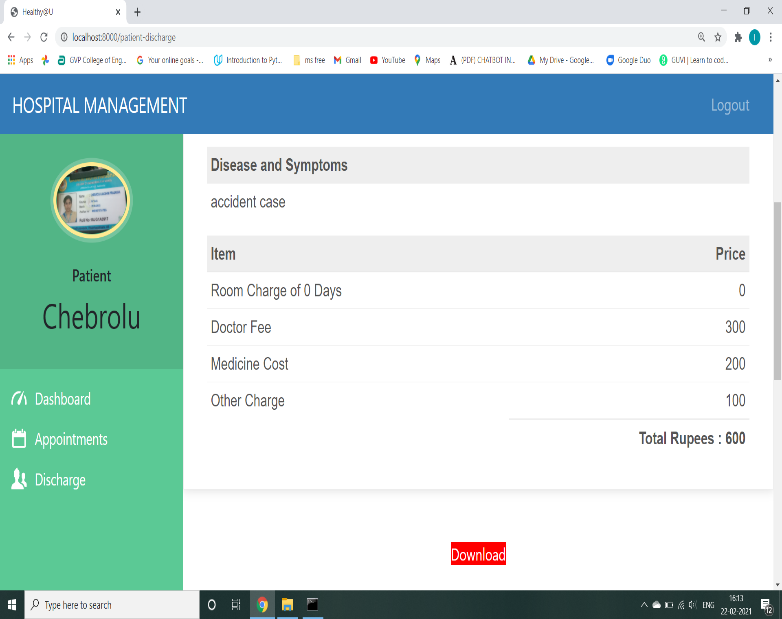
Event cards page:



DoctorDashBoard:



HospitalBill Generation and Download Invoice:



**Conclusion:**

* Chatbots offer an engaging way to communicate with patients and provide them with timely information.
* Chatbot offer numerous advantages for healthcare providers when done right. They can reduce costs and streamline other administrative tasks.
* Appointment booking chatbot manages booking, rescheduling, and cancellation of the appointment with ease so that your employee no longer needs to be tied up.

**References:**

[1] [www.geeksforgeeks](http://www.geeksforgeeks/)

[2] <https://djangobook.com/mdj2-modles>

[3] [https://stackoverflow.com](https://stackoverflow.com/)

[4] <https://docs.djangoproject.com/en/3.1/intro/tutorial/>

[#database=setup](https://docs.djangoproject.com/en/3.1/intro/tutorial/)

[5] [https://developer.mozilla.org/en-US/docs/Learn/Server-side/](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Forms)

[Django/Forms](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Forms)