BLOOD BANK MANAGEMENT PROJECT

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

Rajiv Gandhi University of Knowledge Technologies
R.K.VALLEY

Submitted by:-

K.Mounasree	(R170627)
J.Dhanalakshmi	(O170357)
S.Sravya	(R170810)

Under the Esteemed guidance of Ms.V.Sravani

Department of Computer Science and Engineering

RGUKT RK Valley

Rajiv Gandhi University of Knowledge Techonolgies

RK Valley, Kadapa (Dist), Andhra Pradesh, 516330

CERTIFICATE

This is to certify that the project work titled “**BLOOD BANK MANAGEMENT**” is a bonafied project work submitted by **K.Mounasree, J.Dhanalakshmi and S.Sravya** in **COMPUTER SCIENCE AND ENGINEERING** in partial fulfillment of requirements for the award of degree of **Bachelor of Technology** for the year **2022-2023**carried out the work under the supervision.

INTERNAL GUIDE	**HEAD OF THE DEPARTMENT**
(V SRAVANI)	(N SATYANANDARAM)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success.

I am extremely grateful to our respected Director,Prof. K. SANDHYA RANI for fostering an excellent academic climate in our institution.

I also express my sincere gratitude to our respected Head of the Department Mr SATYANANDARAM sir for his encouragement, overall guidance in viewing this project a good asset and effort in bringing out this project.

I would like to convey thanks to our guide at college V SRAVANI for her guidance, encouragement, co-operation and kindness during the entire duration of the course and academics.

My sincere thanks to all the members who helped me directly and indirectly in the completion of project work. I express my profound gratitude to all our friends and family members for their encouragement.

INDEX

ABSTRACT

Blood Bank Management System (BBMS) is a browser based system that is designed to store, process, retrieve and analyze information concerned with the admin within a blood bank. This project aims at all the information pertaining to blood donors, different blood groups available in each blood bank and help them manage in a better way. Aim is to provide transparency in this field, make the process of obtaining blood from a blood bank hassle free and corruption free and make the system of blood bank management effective. The donors who are interested in donating blood has to register in the database. The requirement of the blood has to be requested and we supply the Blood. Then blood unitsstatus is updated whether they are available or not.

1. Introduction

1.1 Problem Statement

On the day following a blooddonation, the Blood Bank Testing Unit tests all blood for blood type and potential viral agents. They send the results of these tests to the Blood Inventory (another unit of the Centre). For each tested blood unit, if the tests indicate that the blood may be contaminated with a viral agent, the blood unit is destroyed. This is indicated on the test form. The Blood Bank distributes blood to various patients requesting blood. Requests usually come in for specific blood types. The Blood Bank prepares refrigerated containers of these units and distributes them to the patient when they place the order. The BloodBank receives request specific units of blood to supply to the patient from the Blood Bank. When the order is filled, the Blood Bank Manager accepts the order and sends blood to the Patients.

1.2 Purpose

The Blood Bank Management system is a great project. This project is designed for successful completion of a project on blood bank management system.The basic building aim is to provide blood donation service to the city recently. Blood Bank Management system is a web- based application that is designed to store, process, retrieve and analyse information concerned with the administrative and inventory management within a Blood Bank. This project aims at maintaining all the information pertaining to blood donors, different blood groups available in every Blood Bank and help them manage ina better way.Project aim is to provide transparency in this field, make the process of obtaining blood from a Blood Bank hassle-free and corruption-free and make the system of Blood Bank Management effective.

1.3 Scope

The specification builds on the experience of IT technology in blood transfusion that is currently available and informs both Connecting for Health (CfH) and commercial companies producing both hardware and software. The main objective of this specification is to support theautomated tracking of blood products from the initial collection of the blood unit to the final ordering and purchase of the units by hospitals.

Technologies

1.HTML(HYPERTEXT MARKUP LANGUAGES)

2.CSS(CASCADING STYLE SHEET)

3.BOOTSTRAP

4.DJANGO

2. Overall Description

2.1 Product Perspective

This system includes online components. The collection of blood will be manual through Blood Bank. The donor can either register on the Blood Bank website on his own or can visit the Blood Bank An online database is maintained with all the information about the donors. Once the blood is collected it stored in a safe place. An online Blood Inventory Database is maintained as well for the Blood Units collected.

The blood samples are tested to determine whether they are fit to be used or not..Hospitals place orders from this Blood Bank. A record of the order and payment is maintained by both parties. Once the order is placed the Blood Bank Manager send the receipt. Once the hospital makes the payment the order is delivered by the Blood Bank staff.

2.2 Product Function

According to this product, a Donors can create an account back at home or register themselves at the spot of blood camp before donating blood. The Hospital Manger has to register the Hospital which act as an acceptor here. The details of the blood inventory i.e., the availability of a particular type of blood is regularly updated and maintained by the ADMIN. It is a confidential data so the access is only with the administrators.

The registered hospital can place an online order. The order is processed by the Inventory Manager who can check the database of the blood units. If the required blood type and the amount is available, it notifies the corresponding hospitals. When the Hospital Manager confirms the Order, the details are being sent to it.

2.3 User Characteristics

There are mainly three users interacting with each other in this system: Donor, Patient, Admin.

The Donors register through the website or at the spot of blood donation.
The Admin assists them and maintain the details of the Donors. The Admin updates the donor database.

The Admin who collects the blood units from the donors with proper procedure. After the collection and packaging of the blood. The Doctor sends the samples to the Lab Technician for Blood testing and the rest of the blood units to the blood inventory. The Lab Technician carries out test on each blood sample.

He sends the blood report to the Inventory Manager. The Admin discards the unfit blood and store the healthy blood in the respective places. Then he/she updates the required changes in the blood inventory database.

He/she keeps a track of the date of the blood was newly stored in the inventory and its expiry date.
The Patient places an order for the blood units as per requirement. The Admin go throughs the order.
Admin checks and maintains the details.

2.4 Constraints

The Donor and the acceptor are constrained to create an account first to avail the services.The internet connection is also a constraint for this web application.The web application is also constrained by the database capacity so it works well with a smaller number of donors . The access to manage the databases are different for different people. The Admin is given the access to maintain the database of the registered donors and Patients.

2.5 Assumptions and Dependencies

It is assumed that the users have enough resources to run the web application i.e a mobile phone or a computer that supports the required functions.

The web application uses Django for creating and managing the database.
The front end is designed with the help of HTML and CSS.

3. Requirements

3.1 Functional Requirements

1. Access Website:

User should be able to access web-application through either an application browser or similar service on the mobile phone or computer. There should not be any limitation to access web-application.

2. User Registration:

Given that user has accessed web-application, then the user should be able to register through the web-application. The donor user must provide first name, gender, blood group, location, contact , username and password.

3. User log-in:

Given that the user has registered, then the user should be able to login to the web-application. The login information will be stored on the database for future use.

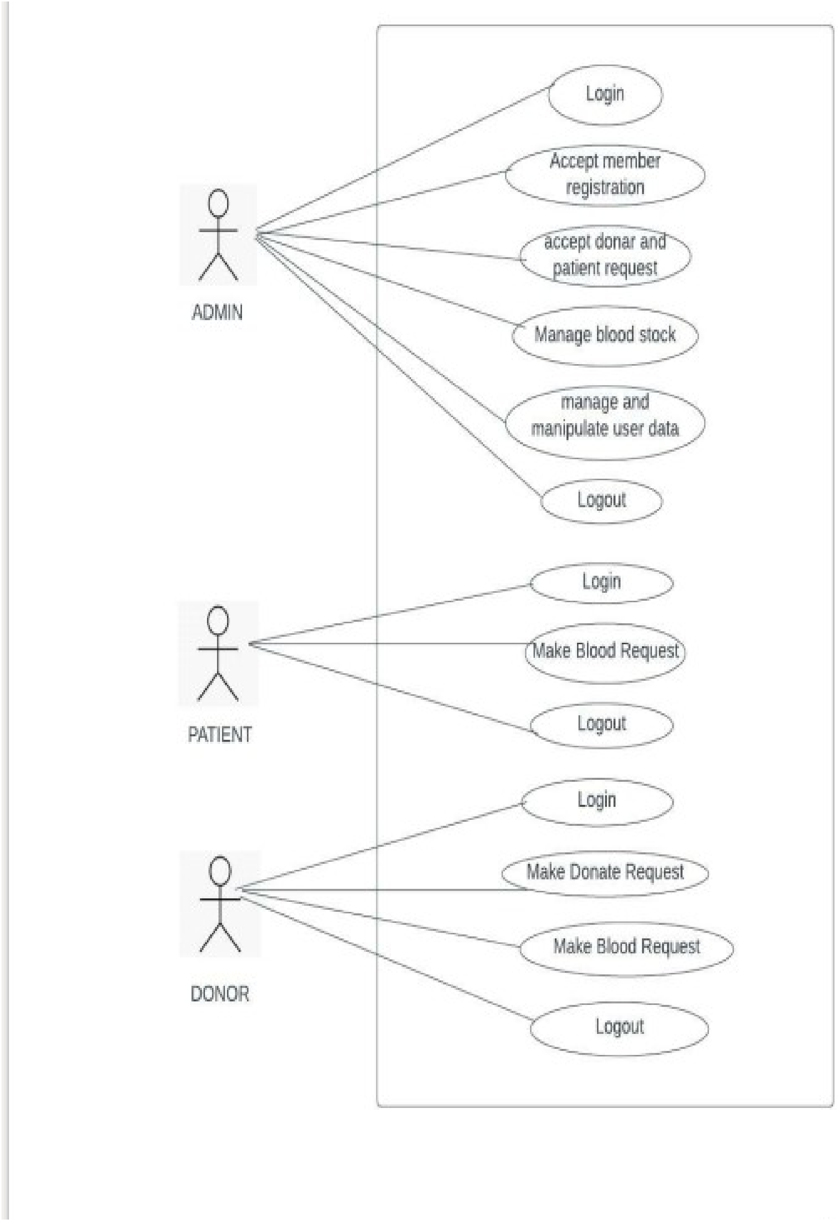
4. Request Blood:

User(Patient) should be able to request for blood at emergency situation, user need to define blood group, location, required date, contact. The order requested will be sent to blood bank and then to the Inventory to check the availability. If available, the requested blood will be sent to the requested donor(Hospital).

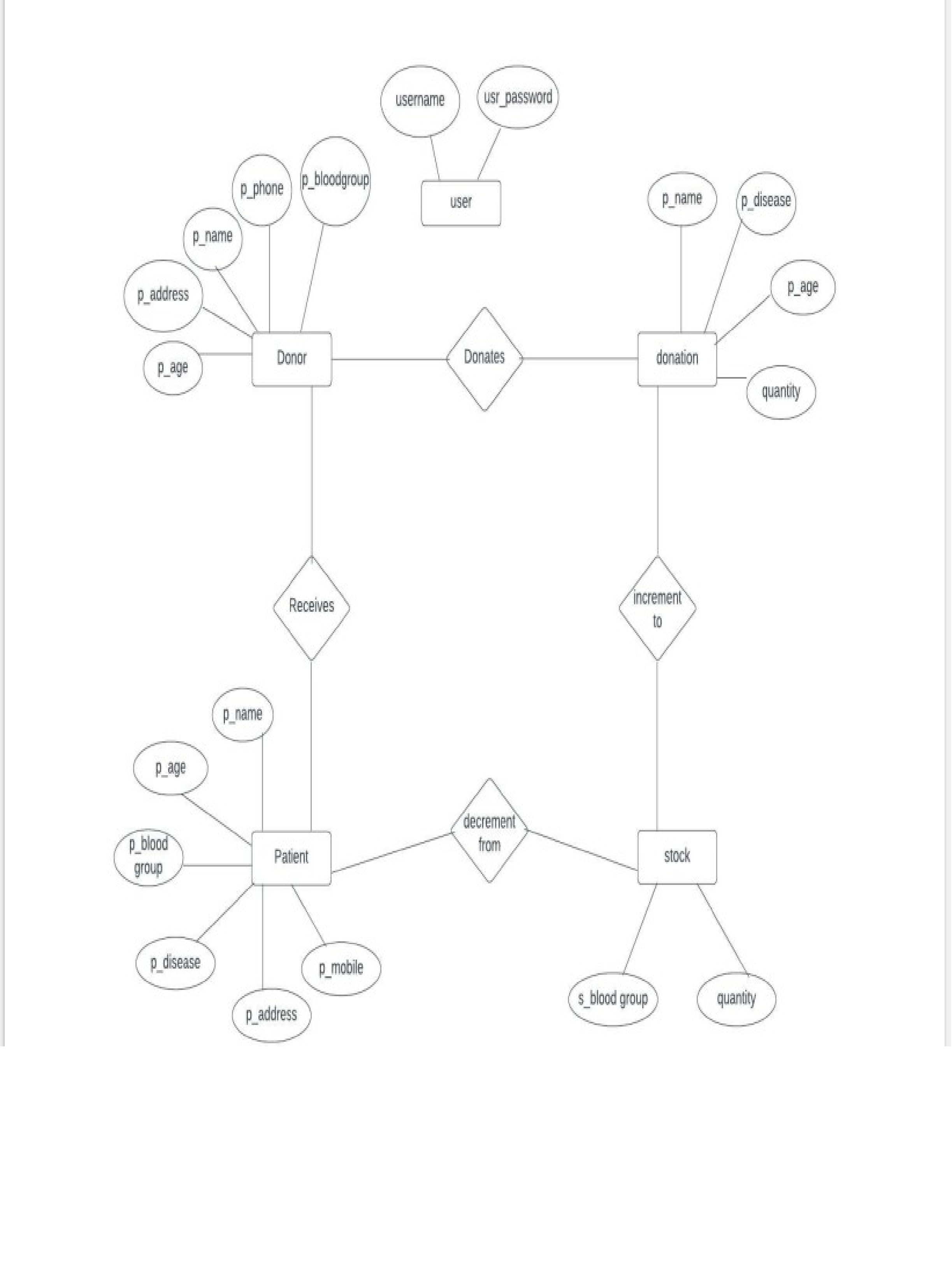
5. View Request:

The Blood Bank should be able to view received request and then respond to them and can search requests by selecting two options select blood group and provision.

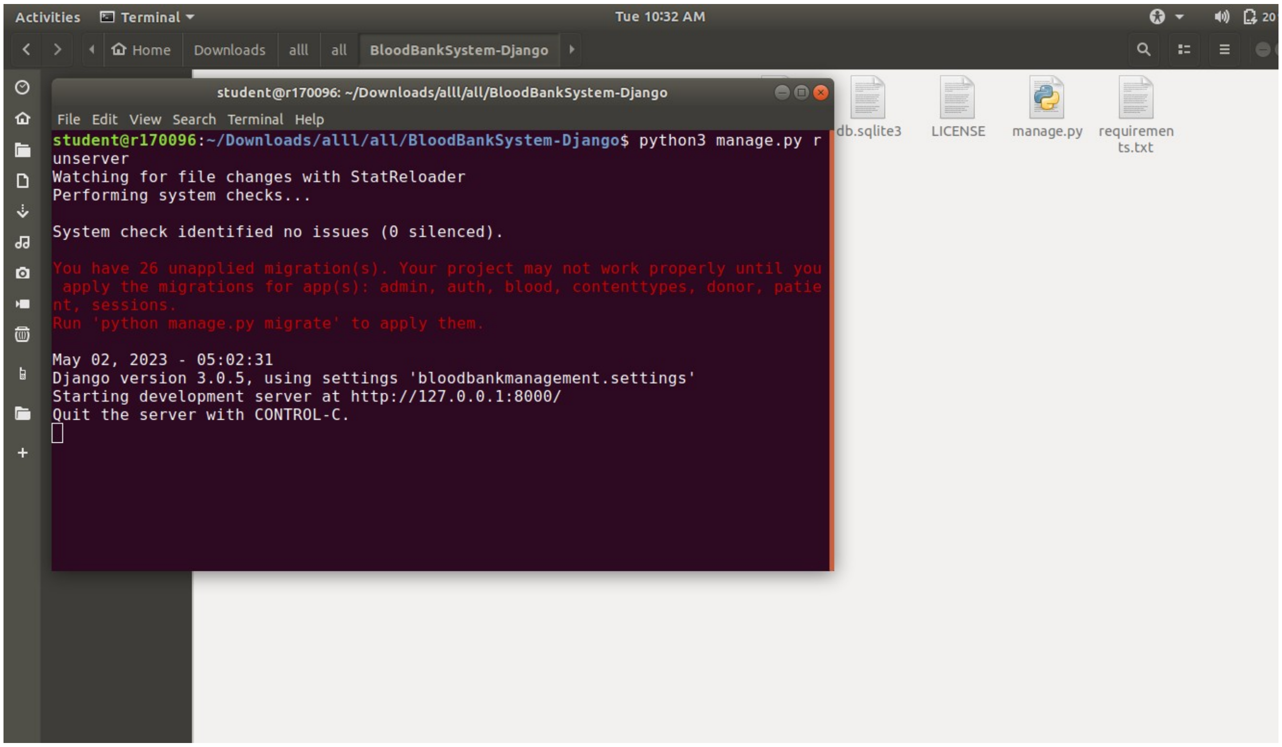
4.Usecase Diagram:



5.ER DIAGRAM:


username
usr_password
user
p_phone
p_bloodgroup
p_name
p_address
p_age
Donor
Donates
donation
p_name
p_disease
p_age
quantity
Receives
increment to
p_name
p_age
p_blood group
Patient
p_disease
p_address
p_mobile
decrement from
stock
s_blood group
quantity

6.CODE SNIPPETS:



```
student@r170096:~/Downloads/alll/all/BloodBankSystem-Django$ python3 manage.py r
unserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 26 unapplied migration(s). Your project may not work properly until you
 apply the migrations for app(s): admin, auth, blood, contenttypes, donor, patie
nt, sessions.
Run 'python manage.py migrate' to apply them.

May 02, 2023 - 05:02:31
Django version 3.0.5, using settings 'bloodbankmanagement.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

URLS.PY

```
"""bloodbankmanagement URL Configuration


"""
from django.contrib import admin
from django.urls import path,include
from django.contrib.auth.views import LogoutView,LoginView
from blood import views
urlpatterns = [
    path('admin/', admin.site.urls),



    path('donor/',include('donor.urls')),
    path('patient/',include('patient.urls')),
    path('',views.home_view,name=''),
    path('logout', LogoutView.as_view(template_name='blood/logout.html'),name='logout'),
path('afterlogin', views.afterlogin_view,name='afterlogin'),
    path('adminlogin',
LoginView.as_view(template_name='blood/adminlogin.html'),name='adminlogin'),
    path('admin-dashboard', views.admin_dashboard_view,name='admin-dashboard'),
    path('admin-blood', views.admin_blood_view,name='admin-blood'),
    path('admin-donor', views.admin_donor_view,name='admin-donor'),
    path('admin-patient', views.admin_patient_view,name='admin-patient'),
    path('update-donor/<int:pk>', views.update_donor_view,name='update-donor'),
    path('delete-donor/<int:pk>', views.delete_donor_view,name='delete-donor'),
    path('admin-request', views.admin_request_view,name='admin-request'),
    path('update-patient/<int:pk>', views.update_patient_view,name='update-patient'),
    path('delete-patient/<int:pk>', views.delete_patient_view,name='delete-patient'),
    path('admin-donation', views.admin_donation_view,name='admin-donation'),
    path('approve-donation/<int:pk>', views.approve_donation_view,name='approve-donation'),
    path('reject-donation/<int:pk>', views.reject_donation_view,name='reject-donation'),
    path('admin-request-history', views.admin_request_history_view,name='admin-request-history'),
    path('update-approve-status/<int:pk>', views.update_approve_status_view,name='update-approve-status'),
    path('update-reject-status/<int:pk>', views.update_reject_status_view,name='update-reject-status'),

]
```

PATIENT _VIEWS.PY

```
from django.shortcuts import render,redirect,reverse
from . import forms,models
from django.db.models import Sum,Q
from django.contrib.auth.models import Group
from django.http import HttpResponseRedirect
from django.contrib.auth.decorators import login_required,user_passes_test
from django.conf import settings
from datetime import date, timedelta
from django.core.mail import send_mail
from django.contrib.auth.models import User
from blood import forms as bforms
from blood import models as bmodels


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.bloodgroup=patientForm.cleaned_data['bloodgroup']
            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,'patient/patientsignup.html',context=mydict)


def patient_dashboard_view(request):
    patient= models.Patient.objects.get(user_id=request.user.id)
    dict={
        'requestpending':
bmodels.BloodRequest.objects.all().filter(request_by_patient=patient).filter(status='Pending').count(
),
        'requestapproved':
bmodels.BloodRequest.objects.all().filter(request_by_patient=patient).filter(status='Approved').cou
nt(),
        'requestmade': bmodels.BloodRequest.objects.all().filter(request_by_patient=patient).count(),
        'requestrejected':
bmodels.BloodRequest.objects.all().filter(request_by_patient=patient).filter(status='Rejected').count
(),

    }

    return render(request,'patient/patient_dashboard.html',context=dict)

def make_request_view(request):
    request_form=bforms.RequestForm()
    if request.method=='POST':
        request_form=bforms.RequestForm(request.POST)
        if request_form.is_valid():
            blood_request=request_form.save(commit=False)
            blood_request.bloodgroup=request_form.cleaned_data['bloodgroup']
            patient= models.Patient.objects.get(user_id=request.user.id)
            blood_request.request_by_patient=patient
            blood_request.save()
            return HttpResponseRedirect('my-request')
__ return render(request,'patient/makerequest.html',{'request_form':request_form})

def my_request_view(request):
    patient= models.Patient.objects.get(user_id=request.user.id)
    blood_request=bmodels.BloodRequest.objects.all().filter(request_by_patient=patient)
    return render(request,'patient/my_request.html',{'blood_request':blood_request})
```

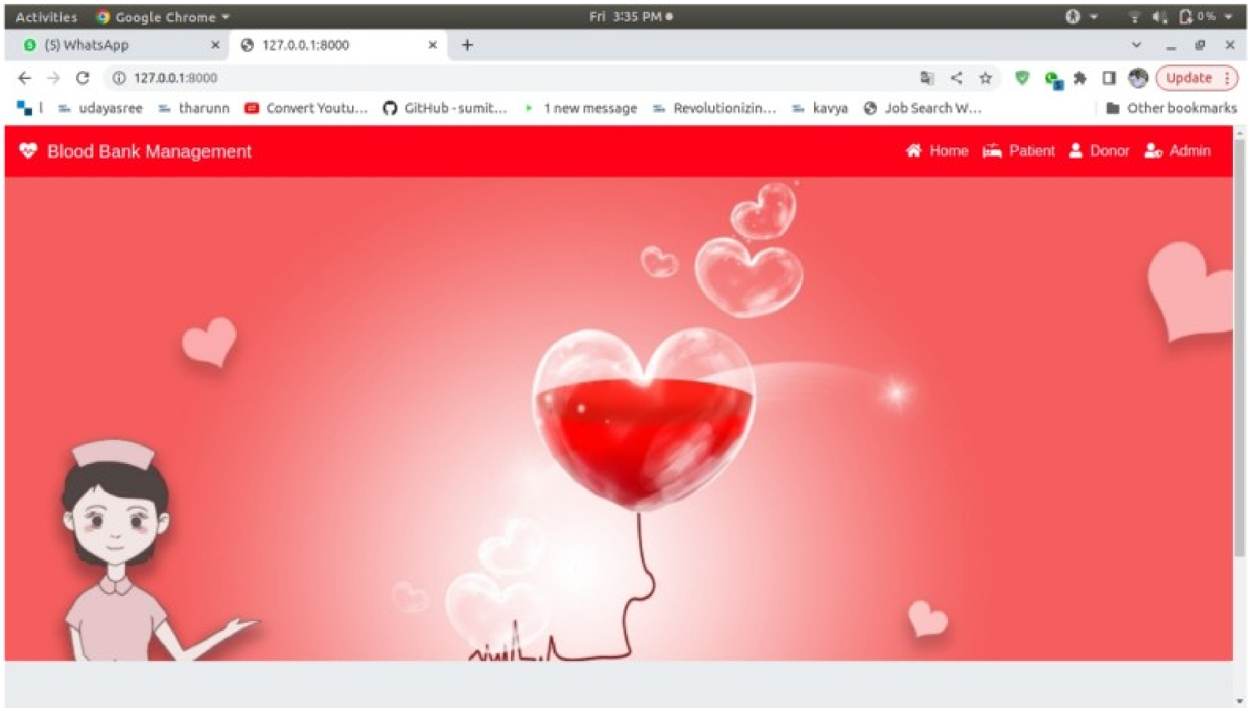
BLOOD_VIEWS.PY

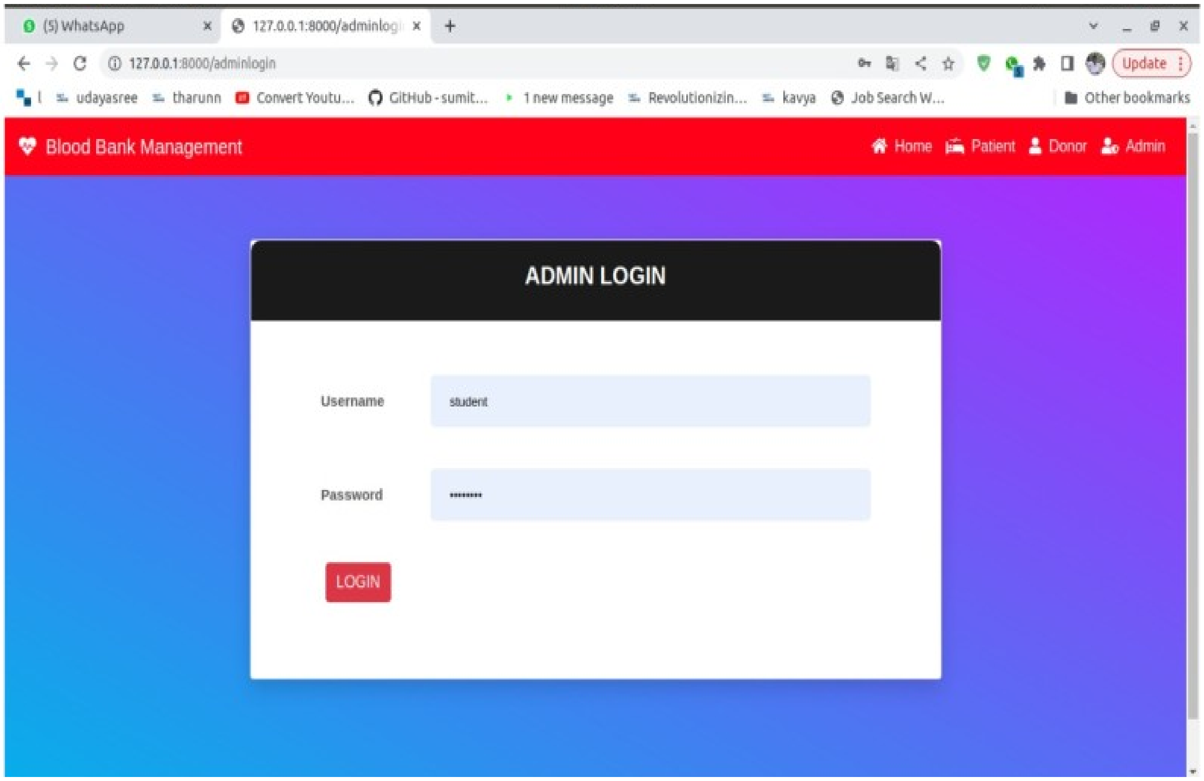
```
from django.shortcuts import render,redirect,reverse
from . import forms,models
from django.db.models import Sum,Q
from django.contrib.auth.models import Group
from django.http import HttpResponseRedirect
from django.contrib.auth.decorators import login_required,user_passes_test
from django.conf import settings
from datetime import date, timedelta
from django.core.mail import send_mail
from django.contrib.auth.models import User
from donor import models as dmodels
from patient import models as pmodels
from donor import forms as dforms
from patient import forms as pforms

def home_view(request):
    x=models.Stock.objects.all()
    print(x)
    if len(x)==0:
        blood1=models.Stock()
        blood1.bloodgroup="A+"
        blood1.save()

        blood3.save()

        blood4=models.Stock()
        blood4.bloodgroup="B-"
        blood4.save()

        blood5=models.Stock()
        blood5.bloodgroup="AB+"
        blood5.save()

        blood6=models.Stock()
        blood6.bloodgroup="AB-"
        blood6.save()

        blood7=models.Stock()
        blood7.bloodgroup="O+"
        blood7.save()

        blood8=models.Stock()
        blood8.bloodgroup="O-"
        blood8.save()

    if request.user.is_authenticated:
        return HttpResponseRedirect('afterlogin')
    return render(request,'blood/index.html')
```

```
def is_donor(user):
    return user.groups.filter(name='DONOR').exists()

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


def afterlogin_view(request):
    if is_donor(request.user):
        return redirect('donor/donor-dashboard')

    elif is_patient(request.user):
        return redirect('patient/patient-dashboard')
    else:
        return redirect('admin-dashboard')

@login_required(login_url='adminlogin')
def admin_dashboard_view(request):
    totalunit=models.Stock.objects.aggregate(Sum('unit'))
    dict={

        'A1':models.Stock.objects.get(bloodgroup="A+"),
        'A2':models.Stock.objects.get(bloodgroup="A-"),
        'B1':models.Stock.objects.get(bloodgroup="B+"),
        'B2':models.Stock.objects.get(bloodgroup="B-"),
        'AB1':models.Stock.objects.get(bloodgroup="AB+"),
        'AB2':models.Stock.objects.get(bloodgroup="AB-"),
        'O1':models.Stock.objects.get(bloodgroup="O+"),
        'O2':models.Stock.objects.get(bloodgroup="O-"),
        'totaldonors':dmodels.Donor.objects.all().count(),
        'totalbloodunit':totalunit['unit__sum'],
        'totalrequest':models.BloodRequest.objects.all().count(),
        'totalapprovedrequest':models.BloodRequest.objects.all().filter(status='Approved').count()
    }
    return render(request,'blood/admin_dashboard.html',context=dict)

@login_required(login_url='adminlogin')
def admin_blood_view(request):
    dict={
        'bloodForm':forms.BloodForm(),
        'A1':models.Stock.objects.get(bloodgroup="A+"),
        'A2':models.Stock.objects.get(bloodgroup="A-"),
        'B1':models.Stock.objects.get(bloodgroup="B+"),
        'B2':models.Stock.objects.get(bloodgroup="B-"),
        'AB1':models.Stock.objects.get(bloodgroup="AB+"),
        'AB2':models.Stock.objects.get(bloodgroup="AB-"),
        'O1':models.Stock.objects.get(bloodgroup="O+"),
        'O2':models.Stock.objects.get(bloodgroup="O-"),
    }
    if request.method=='POST':
        bloodForm=forms.BloodForm(request.POST)
        if bloodForm.is_valid() :
            bloodgroup=bloodForm.cleaned_data['bloodgroup']
            stock=models.Stock.objects.get(bloodgroup=bloodgroup)
            stock.unit=bloodForm.cleaned_data['unit']
            stock.save()
        return HttpResponseRedirect('admin-blood')
    return render(request,'blood/admin_blood.html',context=dict)


@login_required(login_url='adminlogin')
def admin_donor_view(request):
    donors=dmodels.Donor.objects.all()
    return render(request,'blood/admin_donor.html',{'donors':donors})

@login_required(login_url='adminlogin')
def update_donor_view(request,pk):
    donor=dmodels.Donor.objects.get(id=pk)
    user=dmodels.User.objects.get(id=donor.user_id)
    userForm=dforms.DonorUserForm(instance=user)
    donorForm=dforms.DonorForm(request.FILES,instance=donor)
    mydict={'userForm':userForm,'donorForm':donorForm}
    if request.method=='POST':
        userForm=dforms.DonorUserForm(request.POST,instance=user)
        donorForm=dforms.DonorForm(request.POST,request.FILES,instance=donor)
        if userForm.is_valid() and donorForm.is_valid():
            user=userForm.save()
            user.set_password(user.password)
            user.save()
            donor=donorForm.save(commit=False)
            donor.user=user
            donor.bloodgroup=donorForm.cleaned_data['bloodgroup']
            donor.save()
            return redirect('admin-donor')
    return render(request,'blood/update_donor.html',context=mydict)


@login_required(login_url='adminlogin')
def delete_donor_view(request,pk):
    donor=dmodels.Donor.objects.get(id=pk)
    user=User.objects.get(id=donor.user_id)
    user.delete()
    donor.delete()
    return HttpResponseRedirect('/admin-donor')

@login_required(login_url='adminlogin')
def admin_patient_view(request):
    patients=pmodels.Patient.objects.all()
    return render(request,'blood/admin_patient.html',{'patients':patients})


@login_required(login_url='adminlogin')
def update_patient_view(request,pk):
    patient=pmodels.Patient.objects.get(id=pk)
    user=pmodels.User.objects.get(id=patient.user_id)
    userForm=pforms.PatientUserForm(instance=user)
    patientForm=pforms.PatientForm(request.FILES,instance=patient)
    mydict={'userForm':userForm,'patientForm':patientForm}
    if request.method=='POST':
        userForm=pforms.PatientUserForm(request.POST,instance=user)
        patientForm=pforms.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.user=user
            patient.bloodgroup=patientForm.cleaned_data['bloodgroup']
            patient.save()
            return redirect('admin-patient')
    return render(request,'blood/update_patient.html',context=mydict)


@login_required(login_url='adminlogin')
def delete_patient_view(request,pk):
    patient=pmodels.Patient.objects.get(id=pk)
    user=User.objects.get(id=patient.user_id)
    user.delete()
    patient.delete()
    return HttpResponseRedirect('/admin-patient')

@login_required(login_url='adminlogin')
def admin_request_view(request):
    requests=models.BloodRequest.objects.all().filter(status='Pending')
    return render(request,'blood/admin_request.html',{'requests':requests})

@login_required(login_url='adminlogin')
def admin_request_history_view(request):
    requests=models.BloodRequest.objects.all().exclude(status='Pending')
    return render(request,'blood/admin_request_history.html',{'requests':requests})

@login_required(login_url='adminlogin')
def admin_donation_view(request):
    donations=dmodels.BloodDonate.objects.all()
    return render(request,'blood/admin_donation.html',{'donations':donations})
```

```
@login_required(login_url='adminlogin')
def update_approve_status_view(request,pk):
    req=models.BloodRequest.objects.get(id=pk)
    message=None
    bloodgroup=req.bloodgroup
    unit=req.unit
    stock=models.Stock.objects.get(bloodgroup=bloodgroup)
    if stock.unit > unit:
        stock.unit=stock.unit-unit
        stock.save()
        req.status="Approved"

    else:
        message="Stock Doest Not Have Enough Blood To Approve This Request, Only
"+str(stock.unit)+" Unit Available"
    req.save()

    requests=models.BloodRequest.objects.all().filter(status='Pending')
    return render(request,'blood/admin_request.html',{'requests':requests,'message':message})

@login_required(login_url='adminlogin')
def update_reject_status_view(request,pk):
    req=models.BloodRequest.objects.get(id=pk)
    req.status="Rejected"
    req.save()
    return HttpResponseRedirect('/admin-request')

@login_required(login_url='adminlogin')
def approve_donation_view(request,pk):
    donation=dmodels.BloodDonate.objects.get(id=pk)
    donation_blood_group=donation.bloodgroup
    donation_blood_unit=donation.unit
stock=models.Stock.objects.get(bloodgroup=donation_blood_group)
    stock.unit=stock.unit+donation_blood_unit
    stock.save()

    donation.status='Approved'
    donation.save()
    return HttpResponseRedirect('/admin-donation')


@login_required(login_url='adminlogin')
def reject_donation_view(request,pk):
    donation=dmodels.BloodDonate.objects.get(id=pk)
    donation.status='Rejected'
    donation.save()
   return HttpResponseRedirect('/admin-donation')
```

7.PROJECT IMAGES:

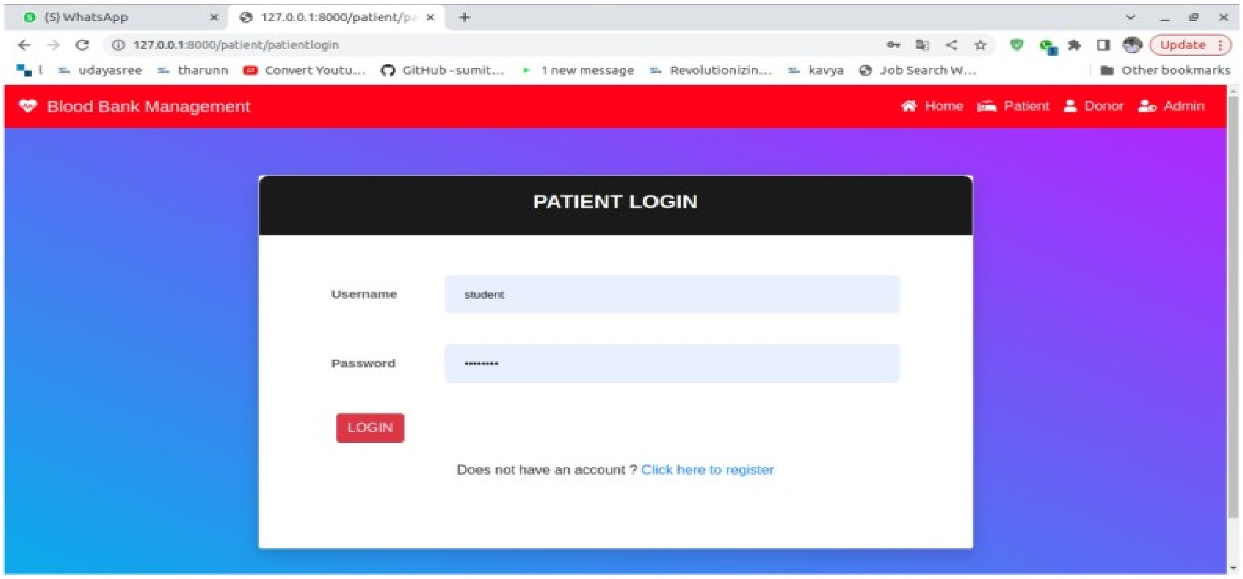




ADMIN LOGIN

Username student

Password ••••••••

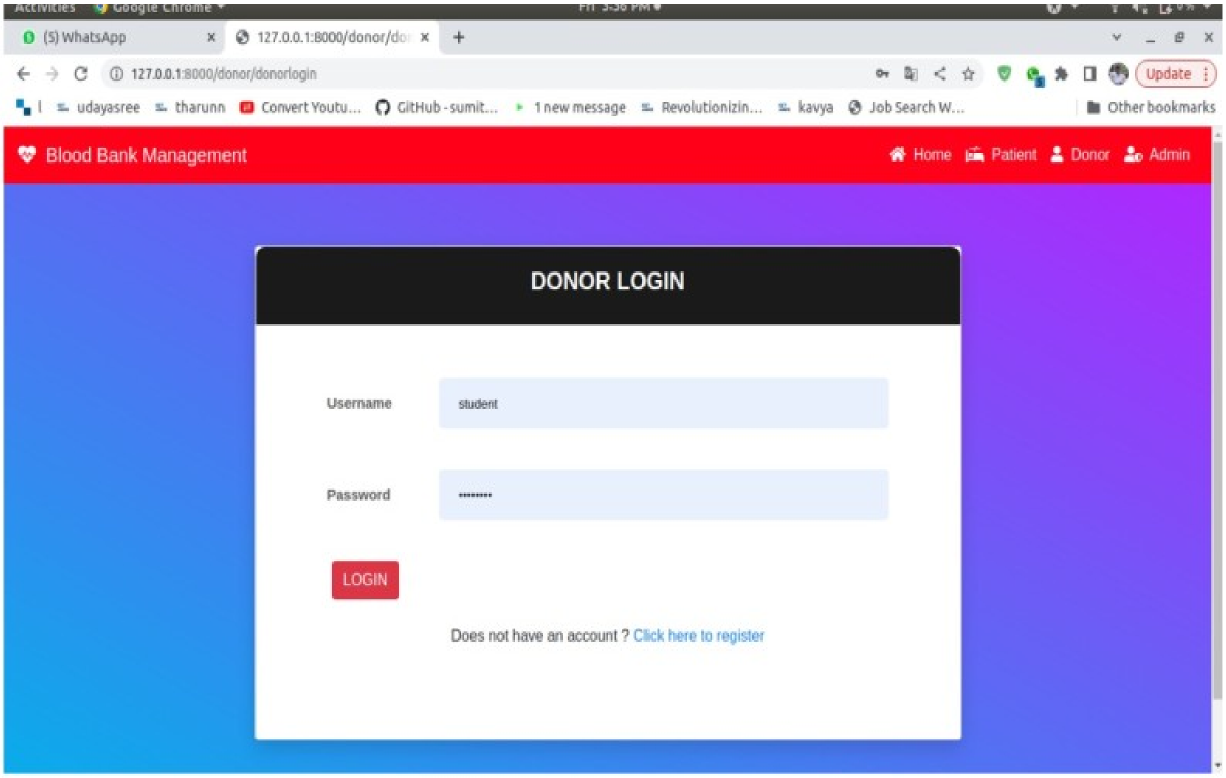


PATIENT LOGIN

Username: student

Password: ••••••••

Does not have an account ? Click here to register

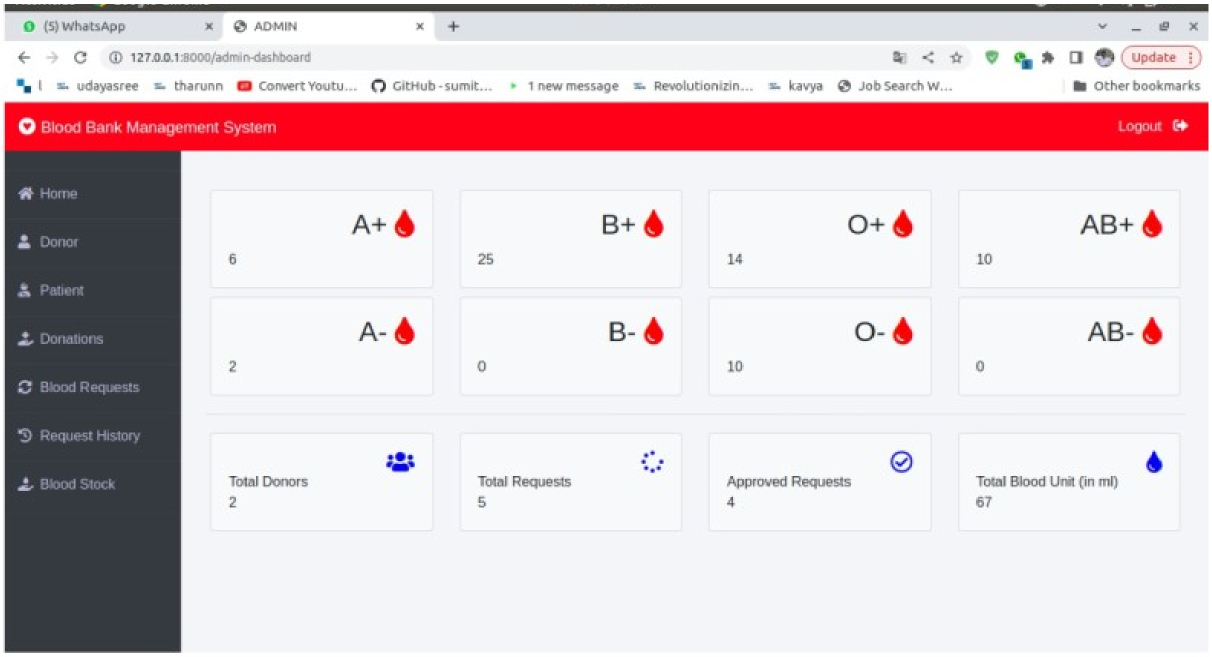


DONOR LOGIN

Username: student

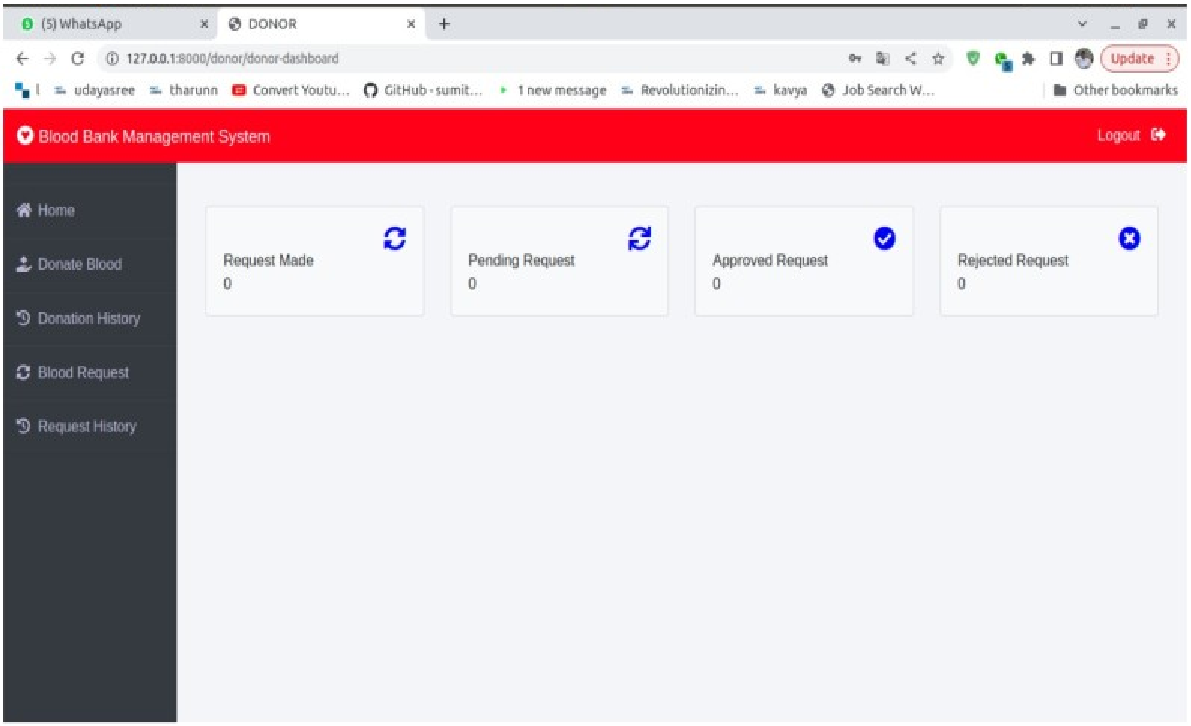
Password: ••••••••

Does not have an account ? Click here to register



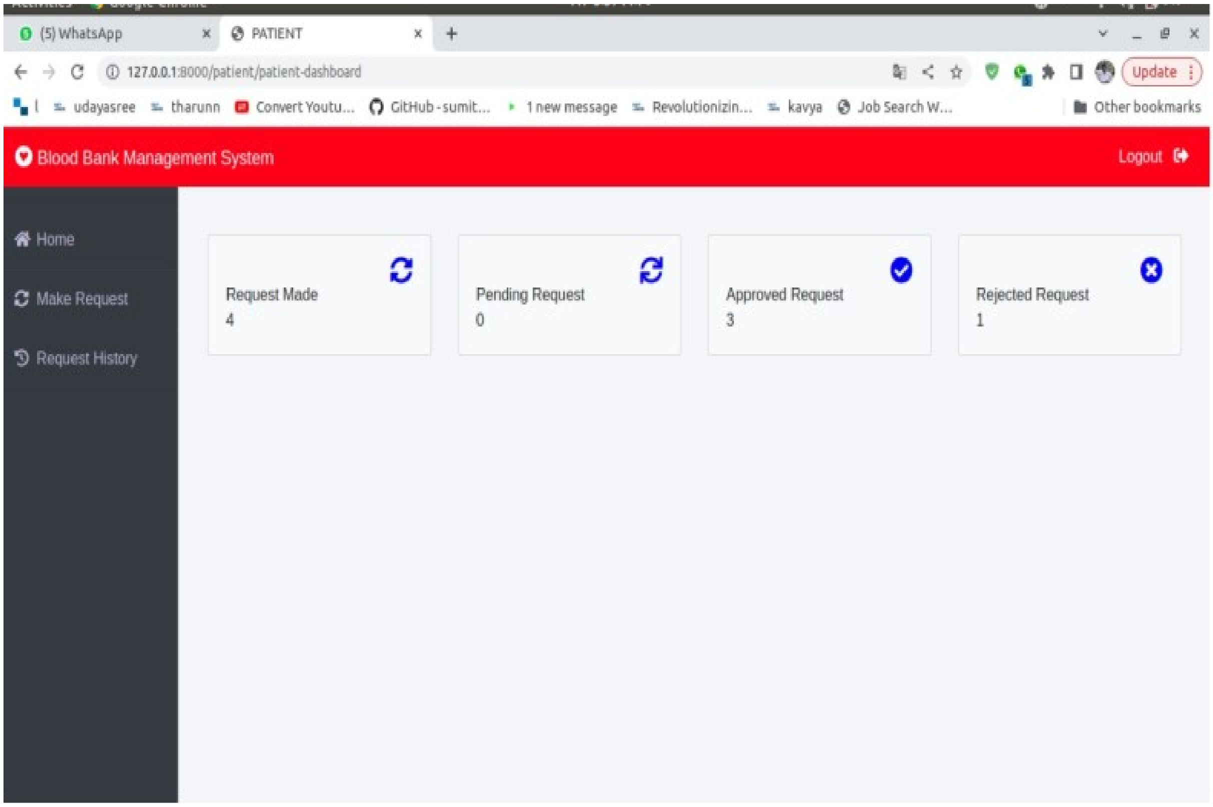
Blood Bank Management System

A+ 6	B+ 25	O+ 14	AB+ 10
A- 2	B- 0	O- 10	AB- 0
Total Donors 2	Total Requests 5	Approved Requests 4	Total Blood Unit (in ml) 67



Blood Bank Management System

Request Made 0	Pending Request 0	Approved Request 0	Rejected Request 0

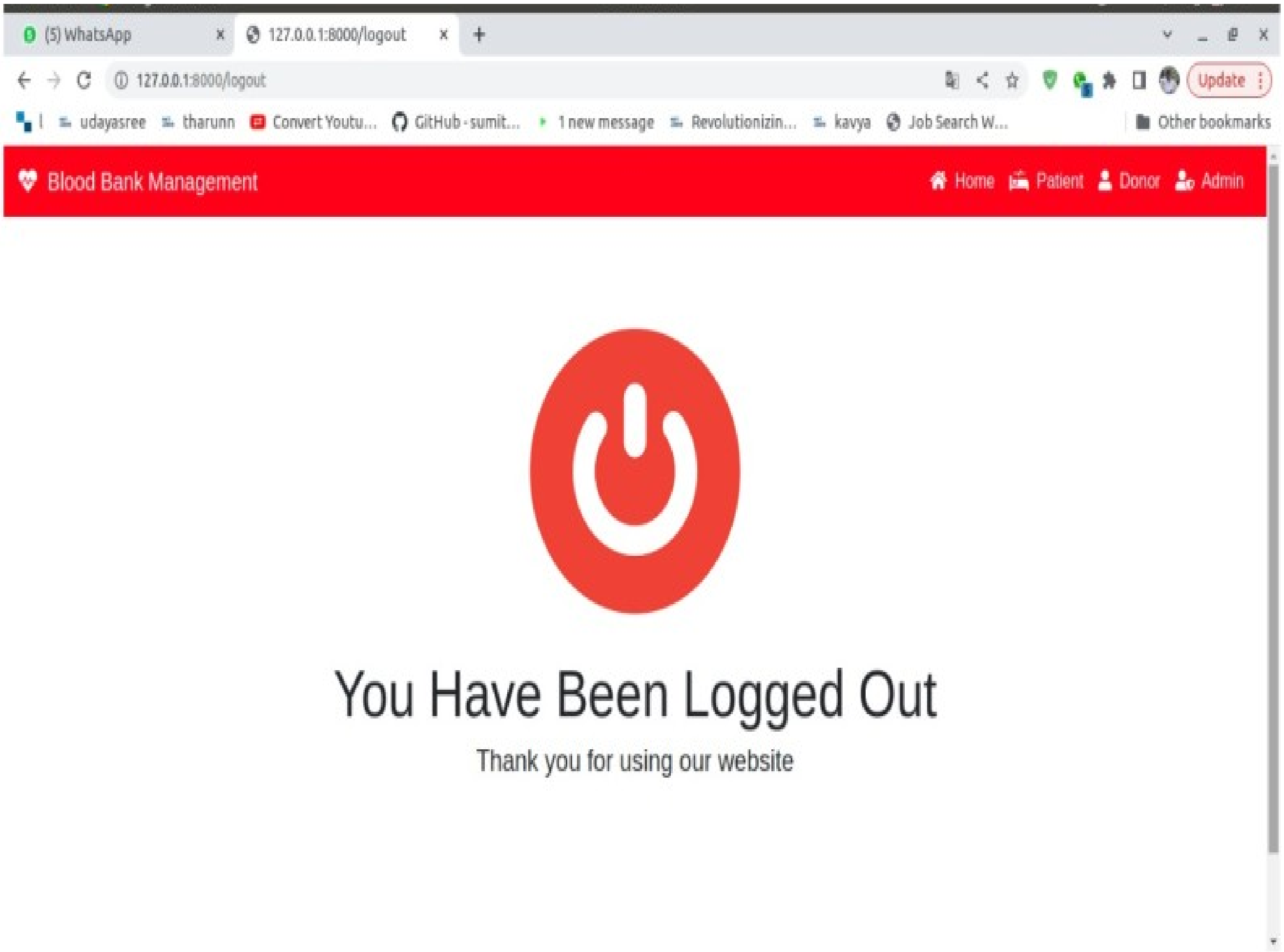


Request Made
4

Pending Request
0

Approved Request
3

Rejected Request
1



You Have Been Logged Out

Thank you for using our website

8.CONCLUSION:

This proposed Blood Bank Management System gives a reliable platform for both donors and acceptors.

The BMMS is a web-based application that helps to minimize human errors and problems pertainingto data redundancy. It is a fast-paced and efficient way tocommunicate without any security threats as the data entered will be verified and frequently updated thereby increasing the probability of saving one's life.

9.REFERENCES:

www.internshala.com

www.github.com

www.youtube.com