**Sarala Birla University**

**Ranchi**



**MAJOR PROJECT REPORT**

**ON**

**BLOOD BANK MANAGEMENT SYSTEM**

**DEPARTMENT OF COMPUTER SCIENCE**

**ENGINEERING**

**PRESENTED  BY:- MENTOR :-**

**Ankit Kumar  (SBU190427) Dr. Partha Paul**

**Dhawan Bhartiya  (SBU190406)**

**Snehal Rani (SBU190408)**

**DECLARATION**

We sincerely declare that Ankit Kumar, Dhawan Bhartiya and Snehal Rani are the sole writers of this Project. The details of our Major project describe our involvement in this project in the field of Computer Science Engineering.

All the information contained in this report is certain and correct to the knowledge.

SIGNATURE: -

NAME: - Ankit Kumar, Dhawan Bhartiya and Snehal Rani

ROLL NO: - BTECH19CSE038, BTECH19CSE017, BTECH19CSE019

DATE: - 08-06-2023

**ACKNOWLEDGEMENT**

We are using this opportunity to express our gratitude to everyone who supported us throughout the course of this Final-Year project on “Blood Bank Management system.”

We are also thankful for their aspiring guidance, invaluably constructive criticism, and friendly advice during the project work. We are sincerely grateful to them for sharing their truthful and illuminating views on several issues related to the project.

We express our warm thanks to **Dr. Partha Paul** for his support and valuable guidance in the process of understanding and realizing the project.

We would also like to thank our friends and all the people who provided us with the facilities being required and conductive conditions for our project.

**ABSTRACT**

The Blood Bank Management System (BBMS) is an application that stores, processes,

retrieves, and analyses data about blood bank administration. It also supervises blood inventory management and other blood bank-related activities. The major goal of the blood bank management system is to keep track of blood, donors, blood groups, blood banks, and stock information. It keeps track of all information concerning blood, blood cells, stocks, and blood.

Because the project is all done at the administrative level, only the administrator can see it. A person who likes to donate blood gives his entire details i.e., fill in the registration form and

can create a username with a password by which he can modify his details if at all there are

any changes in his information given before. Blood is a crucial healthcare resource linked to saving patients' lives with accidents, surgeries, bleeding disorders, pregnancy-related com-

plications, inherited/acquired hematological diseases, and malignancies. Globally, about

118.5 million blood units are collected annually, yet the demand exceeds the existing capacity.

Key words: Online Blood Bank Management System, Blood Bank Management, Blood Donation, Blood Transfusion Safety, Web-Based Application.

**TABLE OF CONTENTS: -**

1. DECLARATION
2. ACKNOWLEDGEMENT
3. ABSTRACT
4. TABLE OF CONTENTS
5. REFERENCES
6. BODY OF PROJECT

* Introduction
* Objective
* Research methodology
* Design process
* System Requirements
* Source Code

7. CONCLUSION

**REFERENCES**

* [Full report on blood bank management system (slideshare.net)](https://www.slideshare.net/JawharAli/full-report-on-blood-bank-management-system)
* [(PDF) Proposal on BLOOD BANK MANAGEMENT SYSTEM | vikrant tickoo - Academia.edu](https://www.academia.edu/28708741/Proposal_on_BLOOD_BANK_MANAGEMENT_SYSTEM)
* [(PDF) Blood Donation Management System | sss sss - Academia.edu](https://www.academia.edu/38786723/Blood_Donation_Management_System)

**INTRODUCTION**

The population of the world is multiplying with each coming year and so are the diseases and health issues. With an increase in the population there is an increase in the need of element in medical science the demand for blood has also increased. Due to the lack of communication between the blood donors and the blood recipients, most of the patients in need of blood do not get blood on time and hence lose their lives. There is a dire need of synchronization between the blood donors and hospitals and the blood banks. This improper management of blood leads to wastage of the available blood inventory. Improper communication and synchronization between the blood banks and hospitals leads to wastage of the blood available. These problems can be dealt with by automating the existing manual blood bank management system. A high-end, efficient, highly available and scalable system must be developed to bridge the gap between the donors and the recipients and to reduce the efforts required to search for blood donors.

**OBJECTIVE**

The main objective of this application is to automate the complete operations of the blood bank. They need to maintain hundreds of thousands of records. Also searching should be very fast so they can find required details instantly. And to develop a web-based portal to facilitate the coordination between supply and demand of blood.

This applied research aims to design, develop and implement online blood bank management system.

This web-based application provides:

• To ensure hospital to have good supply or inventories of blood bags.

• To check the availability of blood bags anytime.

• To manage the information of its blood donor.

• Function to check if the person donate blood for the last 3 months.

• To allow good documentation about the donor and its blood donation activities.

• Support fast searching to find match blood bags for the right person.

This will also serve as the site for interaction of best practices in reducing unnecessary utilization of blood and help the state work more efficiently towards self-sufficiency in blood.

**RESEARCH METHODOLOGY**

1.Project Identification and Selection

In this project, we aimed to develop an online blood bank system which will focus mainly on managing the donor’s blood information. Anyone who is interested in blood donation can donate the blood at the hospital or blood donation centers.

2. Project Initiation and Planning

To begin the project, we have gathered user requirement of this system and prepare the scope and objective. The results from this phase are scope and limitation, objectives, cost and benefits, feature of the proposed system and user interface design.

3.Analyzing System needs

We have studied and identified problems of existing system, then we develop data

flow diagram for the existing system. We also develop data flow diagram (DFD)

and entity relation diagram (E-R diagram) for the proposed system.

4. Designing the Proposed System

Based on the analysis phase, we converted E-R diagram into relational database

model and created data dictionary and DFD and user interface are designed in this

Process.

5. Development of the Proposed System

In this phase, we are going to convert the design of proposed system to computer

software, which includes computer programming using VSCode as a

software tool written in Python, and translating the design specifications into the computer code.

6. Testing the Proposed System

This step is the process of testing whether the programming code will work

correctly with the conditions in our system or not. In this phase, we will fix bugs in

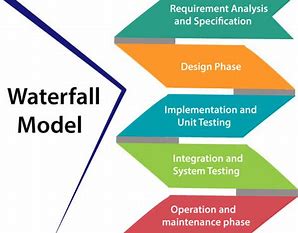
order to produce a system with maximum performance.

7. Implementing the Proposed System

We wish to launch this system on the internet, so that donors can view them

blood donation records online and administrators can create, update, delete, and

query records convenient.



**DESIGN PROCESS**

The User Module

There are two internal users involved in this system. The user requirements are

considered as follows

a. The donor b. The receiver c. The admin

1. The Donor

1. To be able to view their donation records, including where and when they made

donations, and the blood results for each, to learn of their donated blood quality

and schedule their next donations.

2. To be able to view and update their personal information, including name,

contact address, and phone number, to keep their donor’s information record up

to date with the blood bank.

3. To be notified of the blood results of their previous donation by e-mail, to know

the success of their donation.

B. The Receiver

1. To be able to view their donation records, including where and when they receive

donations, and the blood results for each, to learn of their received blood quality.

2. To be able to view and update their personal information, including name,

contact address, and phone number, to keep their receiver’s information record up

to date with the blood bank.

3. To be notified of the blood results of their previous received blood by e-mail, to know

the success.

C. The Admin

1. To be able to create, update, delete, and query donor’s records in order to

manage donor information.

2. To be able to create, update, delete, and retrieve donation records to manage

information about donations made.

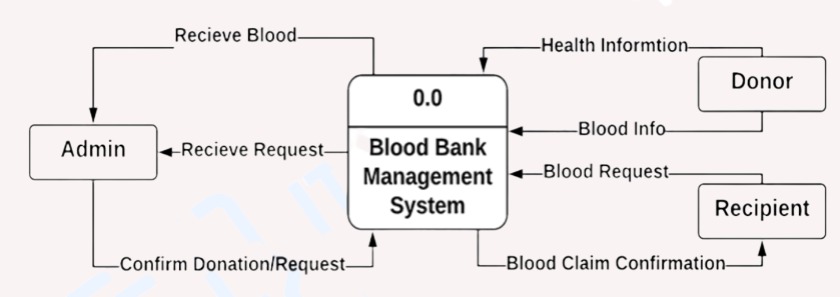
3. To be able to deposit donated blood into inventory when donations are made

**DATA FLOW DIAGRAM**

The flow of data of a system or a process is represented by DFD. It also gives insights of our inputs and outputs of each entity and the process itself. It will help for analyzing existing as well as proposed system.

**DFD Level 0:**

It is also called a Context Diagram. It’s a basic overview of the whole system or process being analyzed or modeled. It’s designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities.

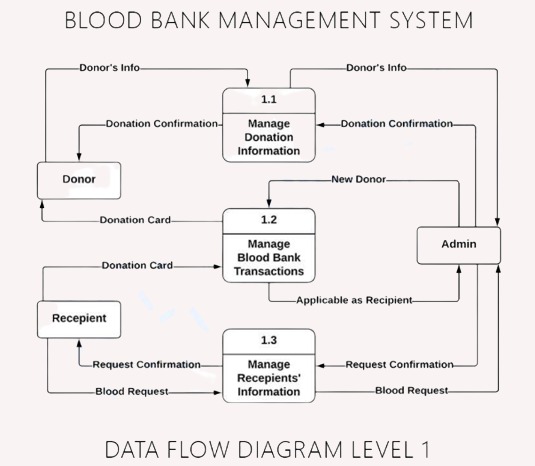


**DFD Level- 1**

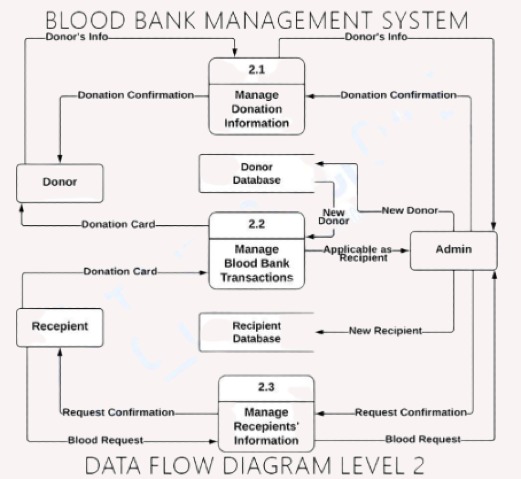
As described previously, context diagrams (level 0 DFDs) are diagrams where the whole system is represented as a single process. A level 1 DFD notates each of the main sub-processes that together form the complete system.

We can think of a level1 DFD as an “exploded view” of the context diagram. A level 1 data flow diagram (DFD) is more detailed than a level 0 DFD as

* It provides a more detailed view of the Context Level Diagram.
* Here, the main functions carried out by the system are highlighted as we break into its sub-processes

****

**DFD LEVEL 2**

****

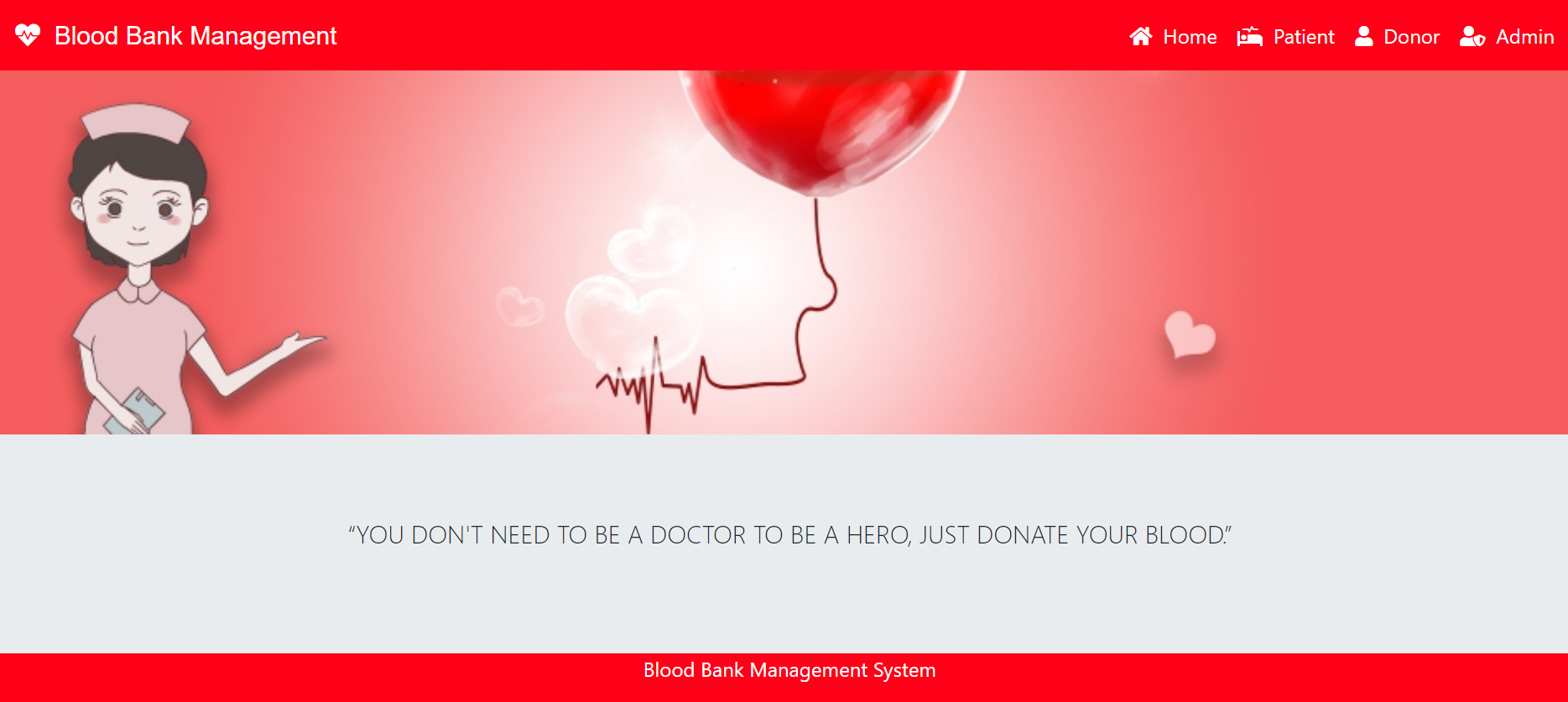
**IMPLEMENTATION**

Step-by-step implementation plan for a Blood Bank Management System:

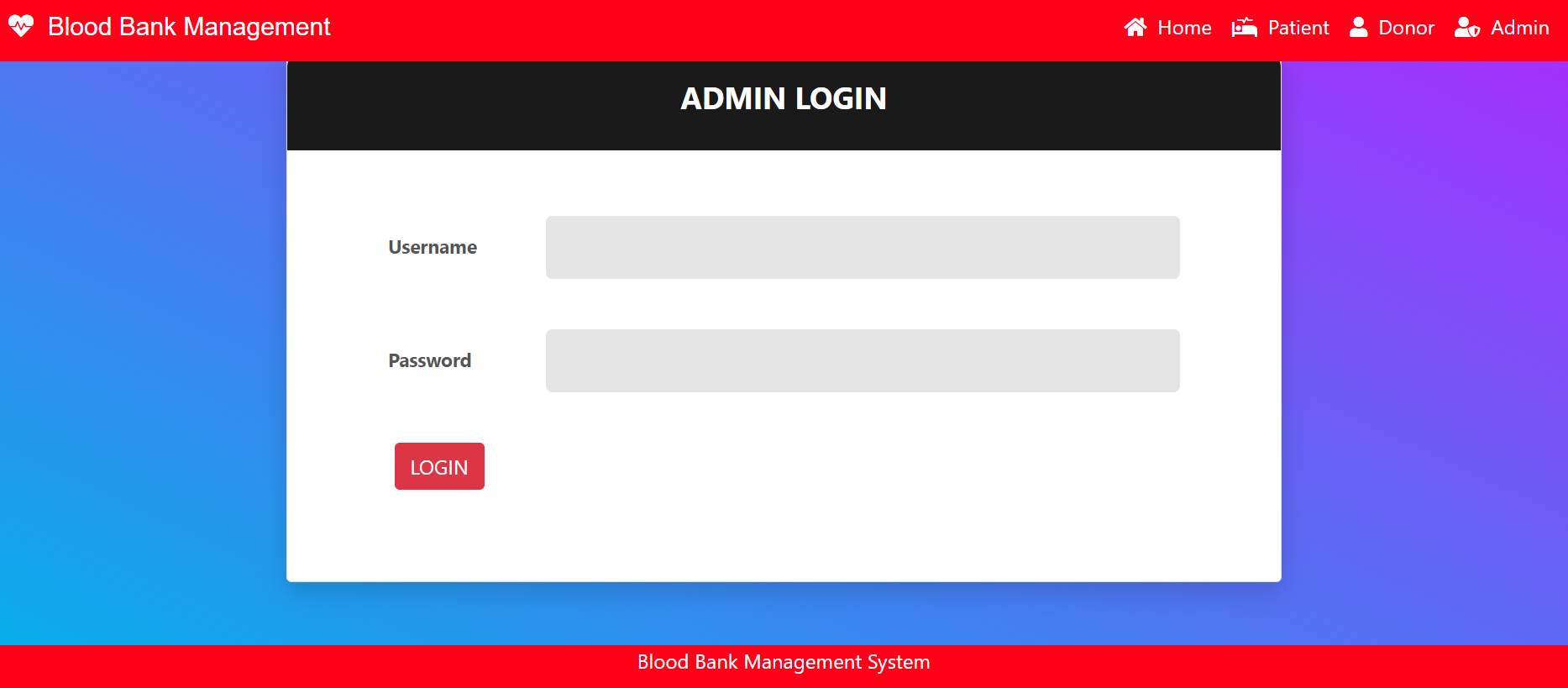
* Define the requirements: Determine the specific needs of the Blood Bank Management System, such as what type of Blood require, what data it needs to track, and what features it needs to have.
* Design the database schema: Create a database schema that includes all the necessary tables and relationships between them. The database schema should be optimized for efficient data retrieval and storage.
* Develop the user interface: Design and develop a user interface that is easy to use and provides all the necessary functionality for managing blood bank. The user interface should be intuitive and user-friendly, with clear navigation and easy access to all features.
* Implement the backend functionality: Develop the backend functionality that handles data storage, retrieval, and manipulation. This should include APIs for communicating with the database and processing user requests.
* Add user authentication and security: Implement user authentication and security measures to protect user data and prevent unauthorized access. This may include password hashing, SSL encryption, and access control measures.
* Test and debug: Test the Blood Bank Management System thoroughly to ensure that it works as expected and there are no bugs or errors. This may include unit testing, integration testing, and user acceptance testing.
* Deploy and maintain: Deploy the Blood Bank Management System to the production environment and monitor it for any issues or errors. Implement a maintenance plan to ensure that the system remains up-to-date and secure, and make any necessary updates or improvements as needed.

**VIEW OF OUR MODEL**

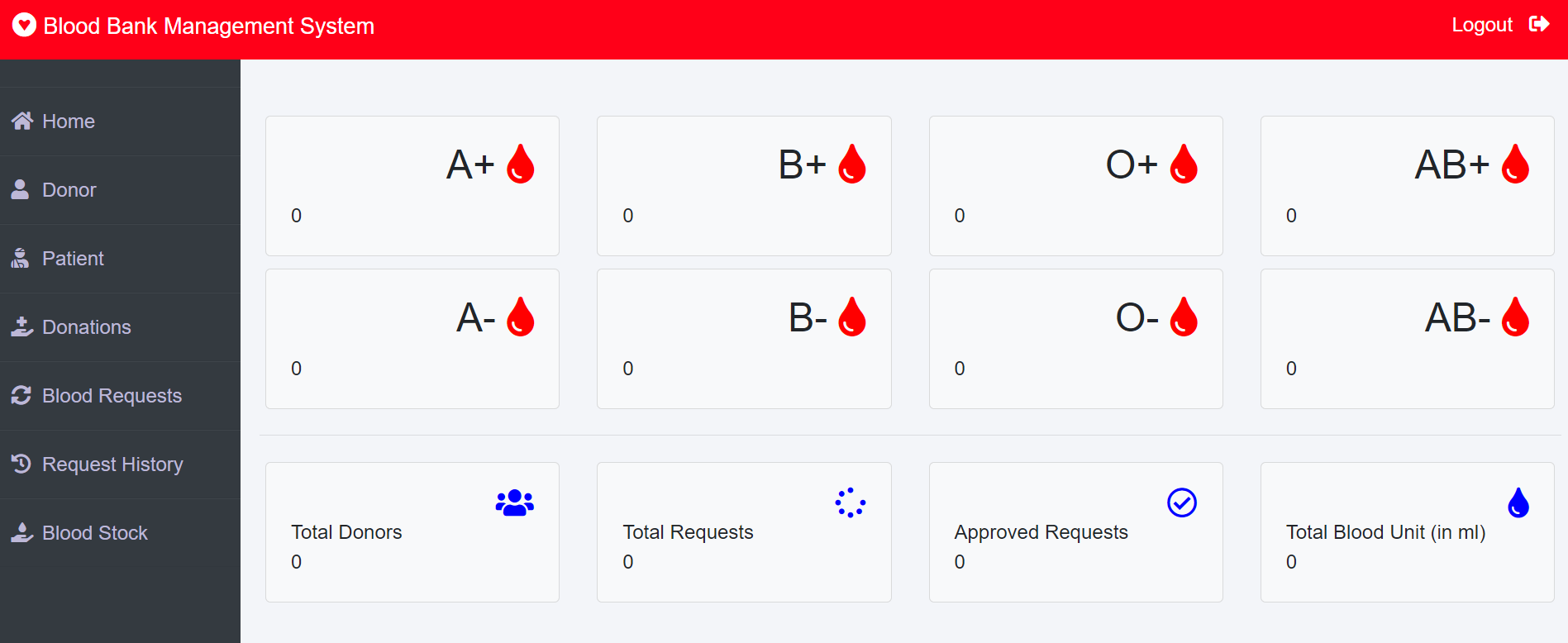
**HOME PAGE:**



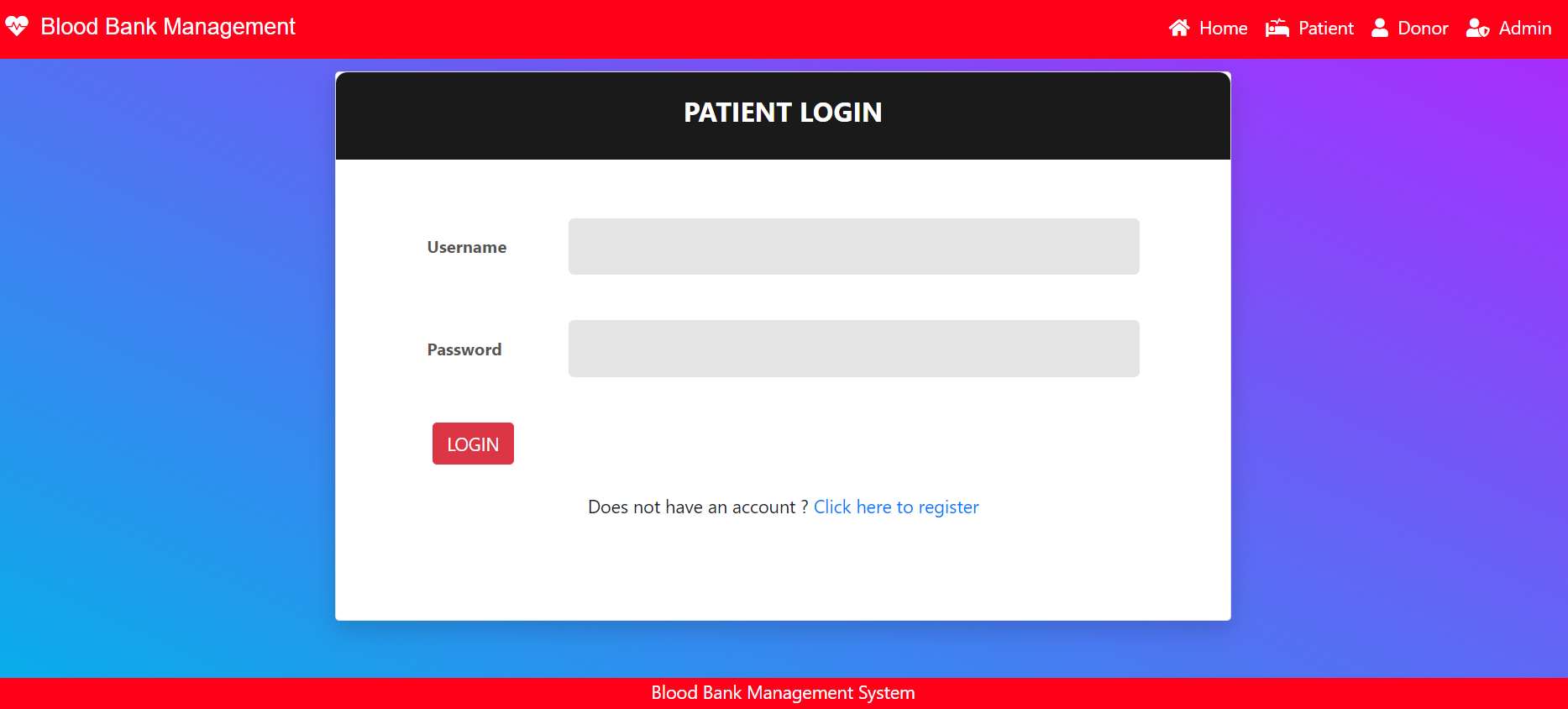
**ADMIN LOGIN PAGE:**



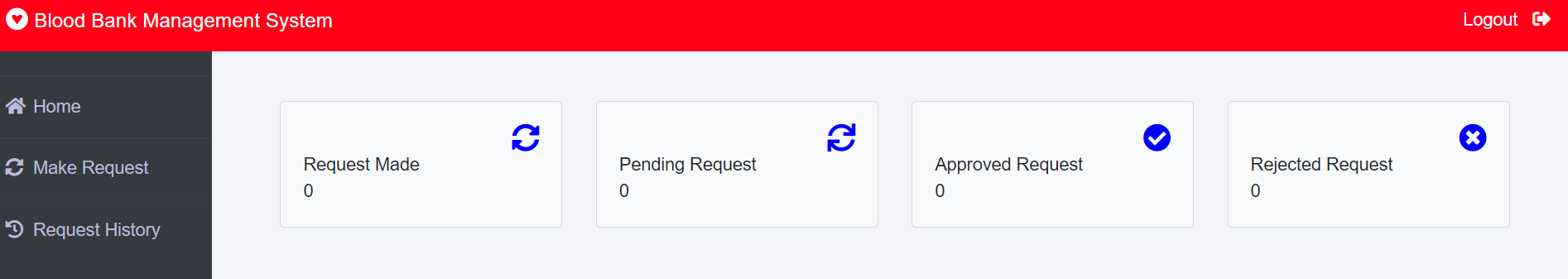
**ADMIN HOME PAGE:**



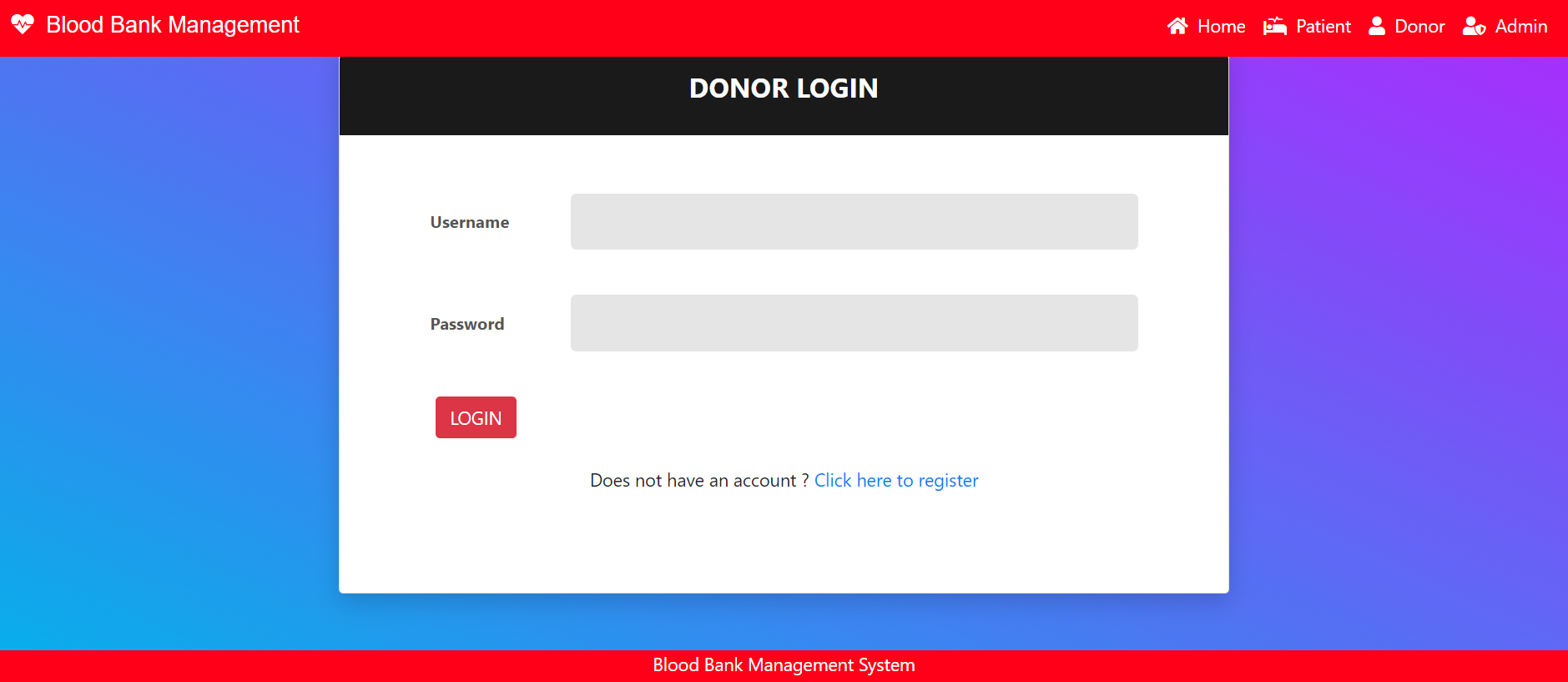
**PATIENT LOGIN PAGE:**



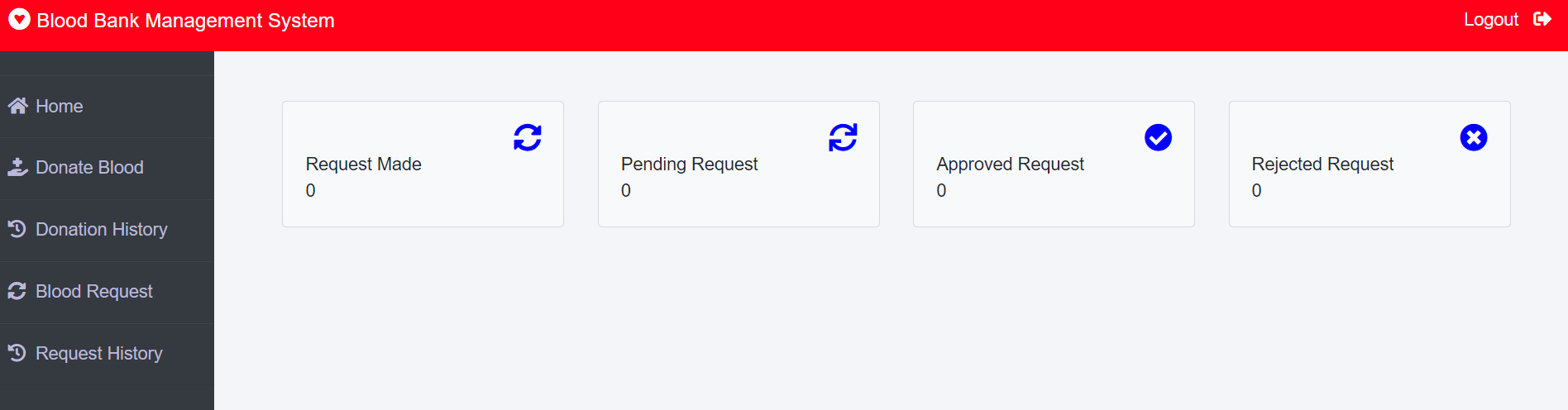
**PATIENT HOME PAGE:**



**DONOR LOGIN PAGE:**



**DONOR HOME PAGE:**



**SYSTEM REQUIREMENTS**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Processor** | **1.9 gigahertz (GHz) x86- or x64-bit dual core processor with SSE2 instruction set** | **3.3 gigahertz (GHz) or faster 64- bit dual core processor with SSE2 instruction set** |
| **Display** | **Super VGA with a resolution of 1024 x 768** | **Super VGA with a resolution of 1024 x 768** |
| **Memory** | **2-GB RAM** | **4-GB RAM or more** |

**LANGUAGE: -**

JAVASCRIPT, CSS, HTML, PYTHON

**DATABASE: -**

MY SQL

**IDE:-**

VISUAL STUDIO

**SOURCE CODE**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* BACKEND CODE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Manage.py**

"""Django's command-line utility for administrative tasks."""

import os

import sys

def main():

    os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'bloodbankmanagement.settings')

    try:

        from django.core.management import execute\_from\_command\_line

    except ImportError as exc:

        raise ImportError(

            "Couldn't import Django. Are you sure it's installed and "

            "available on your PYTHONPATH environment variable? Did you "

            "forget to activate a virtual environment?"

        ) from exc

    execute\_from\_command\_line(sys.argv)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**Donor**

**Admin.py**

from django.contrib import admin

# Register your models here.

**Apps.py**

from django.apps import AppConfig

class DonorConfig(AppConfig):

    name = 'donor'

**forms.py**

from django import forms

from django.contrib.auth.models import User

from . import models

class DonorUserForm(forms.ModelForm):

    class Meta:

        model=User

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

        widgets = {

        'password': forms.PasswordInput()

        }

class DonorForm(forms.ModelForm):

    class Meta:

        model=models.Donor

        fields=['bloodgroup','address','mobile','profile\_pic']

class DonationForm(forms.ModelForm):

    class Meta:

        model=models.BloodDonate

        fields=['age','bloodgroup','disease','unit']

**models.py**

from django.db import models

from django.contrib.auth.models import User

class Donor(models.Model):

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

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

    bloodgroup=models.CharField(max\_length=10)

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

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

    @property

    def get\_name(self):

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

    @property

    def get\_instance(self):

        return self

    def \_\_str\_\_(self):

        return self.user.first\_name

class BloodDonate(models.Model):

    donor=models.ForeignKey(Donor,on\_delete=models.CASCADE)

    disease=models.CharField(max\_length=100,default="Nothing")

    age=models.PositiveIntegerField()

    bloodgroup=models.CharField(max\_length=10)

    unit=models.PositiveIntegerField(default=0)

    status=models.CharField(max\_length=20,default="Pending")

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

    def \_\_str\_\_(self):

        return self.donor

**url.py**

from django.urls import path

from django.contrib.auth.views import LoginView

from . import views

urlpatterns = [

    path('donorlogin', LoginView.as\_view(template\_name='donor/donorlogin.html'),name='donorlogin'),

    path('donorsignup', views.donor\_signup\_view,name='donorsignup'),

    path('donor-dashboard', views.donor\_dashboard\_view,name='donor-dashboard'),

    path('donate-blood', views.donate\_blood\_view,name='donate-blood'),

    path('donation-history', views.donation\_history\_view,name='donation-history'),

    path('make-request', views.make\_request\_view,name='make-request'),

    path('request-history', views.request\_history\_view,name='request-history'),

]

**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 donor\_signup\_view(request):

    userForm=forms.DonorUserForm()

    donorForm=forms.DonorForm()

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

    if request.method=='POST':

        userForm=forms.DonorUserForm(request.POST)

        donorForm=forms.DonorForm(request.POST,request.FILES)

        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()

            my\_donor\_group = Group.objects.get\_or\_create(name='DONOR')

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

        return HttpResponseRedirect('donorlogin')

    return render(request,'donor/donorsignup.html',context=mydict)

def donor\_dashboard\_view(request):

    donor= models.Donor.objects.get(user\_id=request.user.id)

    dict={

        'requestpending': bmodels.BloodRequest.objects.all().filter(request\_by\_donor=donor).filter(status='Pending').count(),

        'requestapproved': bmodels.BloodRequest.objects.all().filter(request\_by\_donor=donor).filter(status='Approved').count(),

        'requestmade': bmodels.BloodRequest.objects.all().filter(request\_by\_donor=donor).count(),

        'requestrejected': bmodels.BloodRequest.objects.all().filter(request\_by\_donor=donor).filter(status='Rejected').count(),

    }

    return render(request,'donor/donor\_dashboard.html',context=dict)

def donate\_blood\_view(request):

    donation\_form=forms.DonationForm()

    if request.method=='POST':

        donation\_form=forms.DonationForm(request.POST)

        if donation\_form.is\_valid():

            blood\_donate=donation\_form.save(commit=False)

            blood\_donate.bloodgroup=donation\_form.cleaned\_data['bloodgroup']

            donor= models.Donor.objects.get(user\_id=request.user.id)

            blood\_donate.donor=donor

            blood\_donate.save()

            return HttpResponseRedirect('donation-history')

    return render(request,'donor/donate\_blood.html',{'donation\_form':donation\_form})

def donation\_history\_view(request):

    donor= models.Donor.objects.get(user\_id=request.user.id)

    donations=models.BloodDonate.objects.all().filter(donor=donor)

    return render(request,'donor/donation\_history.html',{'donations':donations})

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']

            donor= models.Donor.objects.get(user\_id=request.user.id)

            blood\_request.request\_by\_donor=donor

            blood\_request.save()

            return HttpResponseRedirect('request-history')

    return render(request,'donor/makerequest.html',{'request\_form':request\_form})

def request\_history\_view(request):

    donor= models.Donor.objects.get(user\_id=request.user.id)

    blood\_request=bmodels.BloodRequest.objects.all().filter(request\_by\_donor=donor)

    return render(request,'donor/request\_history.html',{'blood\_request':blood\_request})

**PATIENT**

**Admin.py**

from django.contrib import admin

# Register your models here.

**Apps.py**

from django.apps import AppConfig

class PatientConfig(AppConfig):

    name = 'patient'

**forms.py**

from django import forms

from django.contrib.auth.models import User

from . import models

class PatientUserForm(forms.ModelForm):

    class Meta:

        model=User

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

        widgets = {

        'password': forms.PasswordInput()

        }

class PatientForm(forms.ModelForm):

    class Meta:

        model=models.Patient

        fields=['age','bloodgroup','disease','address','doctorname','mobile','profile\_pic']

**models.py**

from django.db import models

from django.contrib.auth.models import User

class Patient(models.Model):

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

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

    age=models.PositiveIntegerField()

    bloodgroup=models.CharField(max\_length=10)

    disease=models.CharField(max\_length=100)

    doctorname=models.CharField(max\_length=50)

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

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

    @property

    def get\_name(self):

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

    @property

    def get\_instance(self):

        return self

    def \_\_str\_\_(self):

        return self.user.first\_name

**urls.py**

from django.urls import path

from django.contrib.auth.views import LoginView

from . import views

urlpatterns = [

    path('patientlogin', LoginView.as\_view(template\_name='patient/patientlogin.html'),name='patientlogin'),

    path('patientsignup', views.patient\_signup\_view,name='patientsignup'),

    path('patient-dashboard', views.patient\_dashboard\_view,name='patient-dashboard'),

    path('make-request', views.make\_request\_view,name='make-request'),

    path('my-request', views.my\_request\_view,name='my-request'),

]

**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').count(),

        '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**

**Admin.py**

from django.contrib import admin

# Register your models here.

**Apps.py**

from django.apps import AppConfig

class BloodConfig(AppConfig):

    name = 'blood'

**forms.py**

from django import forms

from . import models

class BloodForm(forms.ModelForm):

    class Meta:

        model=models.Stock

        fields=['bloodgroup','unit']

class RequestForm(forms.ModelForm):

    class Meta:

        model=models.BloodRequest

        fields=['patient\_name','patient\_age','reason','bloodgroup','unit']

**models.py**

from django.db import models

from patient import models as pmodels

from donor import models as dmodels

class Stock(models.Model):

    bloodgroup=models.CharField(max\_length=10)

    unit=models.PositiveIntegerField(default=0)

    def \_\_str\_\_(self):

        return self.bloodgroup

class BloodRequest(models.Model):

    request\_by\_patient=models.ForeignKey(pmodels.Patient,null=True,on\_delete=models.CASCADE)

    request\_by\_donor=models.ForeignKey(dmodels.Donor,null=True,on\_delete=models.CASCADE)

    patient\_name=models.CharField(max\_length=30)

    patient\_age=models.PositiveIntegerField()

    reason=models.CharField(max\_length=500)

    bloodgroup=models.CharField(max\_length=10)

    unit=models.PositiveIntegerField(default=0)

    status=models.CharField(max\_length=20,default="Pending")

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

    def \_\_str\_\_(self):

        return self.bloodgroup

**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()

        blood2=models.Stock()

        blood2.bloodgroup="A-"

        blood2.save()

        blood3=models.Stock()

        blood3.bloodgroup="B+"

        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')

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FRONTEND CODE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Main.css**

/\* ==========================================================================

   #FONT

   ========================================================================== \*/

.font-robo {

  font-family: "Roboto", "Arial", "Helvetica Neue", sans-serif;

}

.font-poppins {

  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;

}

.font-opensans {

  font-family: "Open Sans", "Arial", "Helvetica Neue", sans-serif;

}

/\* ==========================================================================

   #GRID

   ========================================================================== \*/

.row {

  display: -webkit-box;

  display: -webkit-flex;

  display: -moz-box;

  display: -ms-flexbox;

  display: flex;

  -webkit-flex-wrap: wrap;

  -ms-flex-wrap: wrap;

  flex-wrap: wrap;

}

.row .col-2:last-child .input-group-desc {

  margin-bottom: 0;

}

.row-space {

  -webkit-box-pack: justify;

  -webkit-justify-content: space-between;

  -moz-box-pack: justify;

  -ms-flex-pack: justify;

  justify-content: space-between;

}

.row-refine {

  margin: 0 -15px;

}

.row-refine .col-3 .input-group-desc,

.row-refine .col-9 .input-group-desc {

  margin-bottom: 0;

}

.col-2 {

  width: -webkit-calc((100% - 30px) / 2);

  width: -moz-calc((100% - 30px) / 2);

  width: calc((100% - 30px) / 2);

}

@media (max-width: 767px) {

  .col-2 {

    width: 100%;

  }

}

.form-row {

  display: -webkit-box;

  display: -webkit-flex;

  display: -moz-box;

  display: -ms-flexbox;

  display: flex;

  -webkit-flex-wrap: wrap;

  -ms-flex-wrap: wrap;

  flex-wrap: wrap;

  -webkit-box-align: center;

  -webkit-align-items: center;

  -moz-box-align: center;

  -ms-flex-align: center;

  align-items: center;

  margin-bottom: 40px;

}

.form-row .name {

  width: 125px;

  color: #555;

  font-size: 15px;

  font-weight: 700;

}

.form-row .value {

  width: -webkit-calc(100% - 125px);

  width: -moz-calc(100% - 125px);

  width: calc(100% - 125px);

}

@media (max-width: 767px) {

  .form-row {

    display: block;

  }

  .form-row .name,

  .form-row .value {

    display: block;

    width: 100%;

  }

  .form-row .name {

    margin-bottom: 7px;

  }

}

.col-3,

.col-9 {

  padding: 0 15px;

  position: relative;

  width: 100%;

  min-height: 1px;

}

.col-3 {

  -webkit-box-flex: 0;

  -webkit-flex: 0 0 25%;

  -moz-box-flex: 0;

  -ms-flex: 0 0 25%;

  flex: 0 0 25%;

  max-width: 25%;

}

@media (max-width: 767px) {

  .col-3 {

    -webkit-box-flex: 0;

    -webkit-flex: 0 0 35%;

    -moz-box-flex: 0;

    -ms-flex: 0 0 35%;

    flex: 0 0 35%;

    max-width: 35%;

  }

}

.col-9 {

  -webkit-box-flex: 0;

  -webkit-flex: 0 0 75%;

  -moz-box-flex: 0;

  -ms-flex: 0 0 75%;

  flex: 0 0 75%;

  max-width: 75%;

}

@media (max-width: 767px) {

  .col-9 {

    -webkit-box-flex: 0;

    -webkit-flex: 0 0 65%;

    -moz-box-flex: 0;

    -ms-flex: 0 0 65%;

    flex: 0 0 65%;

    max-width: 65%;

  }

}

/\* ==========================================================================

   #BOX-SIZING

   ========================================================================== \*/

/\*\*

 \* More sensible default box-sizing:

 \* css-tricks.com/inheriting-box-sizing-probably-slightly-better-best-practice

 \*/

html {

  -webkit-box-sizing: border-box;

  -moz-box-sizing: border-box;

  box-sizing: border-box;

}

\* {

  padding: 0;

  margin: 0;

}

\*, \*:before, \*:after {

  -webkit-box-sizing: inherit;

  -moz-box-sizing: inherit;

  box-sizing: inherit;

}

/\* ==========================================================================

   #RESET

   ========================================================================== \*/

/\*\*

 \* A very simple reset that sits on top of Normalize.css.

 \*/

body,

h1, h2, h3, h4, h5, h6,

blockquote, p, pre,

dl, dd, ol, ul,

figure,

hr,

fieldset, legend {

  margin: 0;

  padding: 0;

}

/\*\*

 \* Remove trailing margins from nested lists.

 \*/

li > ol,

li > ul {

  margin-bottom: 0;

}

/\*\*

 \* Remove default table spacing.

 \*/

table {

  border-collapse: collapse;

  border-spacing: 0;

}

/\*\*

 \* 1. Reset Chrome and Firefox behaviour which sets a `min-width: min-content;`

 \*    on fieldsets.

 \*/

fieldset {

  min-width: 0;

  /\* [1] \*/

  border: 0;

}

button {

  outline: none;

  background: none;

  border: none;

}

/\* ==========================================================================

   #PAGE WRAPPER

   ========================================================================== \*/

.page-wrapper {

  min-height: 100vh;

}

body {

  font-family: "Open Sans", "Arial", "Helvetica Neue", sans-serif;

  font-weight: 400;

  font-size: 14px;

}

h1, h2, h3, h4, h5, h6 {

  font-weight: 400;

}

h1 {

  font-size: 36px;

}

h2 {

  font-size: 30px;

}

h3 {

  font-size: 24px;

}

h4 {

  font-size: 18px;

}

h5 {

  font-size: 15px;

}

h6 {

  font-size: 13px;

}

/\* ==========================================================================

   #BACKGROUND

   ========================================================================== \*/

.bg-blue {

  background: #2c6ed5;

}

.bg-red {

  background: #fa4251;

}

.bg-gra-01 {

  background: -webkit-gradient(linear, left bottom, left top, from(#fbc2eb), to(#a18cd1));

  background: -webkit-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);

  background: -moz-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);

  background: -o-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);

  background: linear-gradient(to top, #fbc2eb 0%, #a18cd1 100%);

}

.bg-gra-02 {

  background: -webkit-gradient(linear, left bottom, right top, from(#fc2c77), to(#6c4079));

  background: -webkit-linear-gradient(bottom left, #fc2c77 0%, #6c4079 100%);

  background: -moz-linear-gradient(bottom left, #fc2c77 0%, #6c4079 100%);

  background: -o-linear-gradient(bottom left, #fc2c77 0%, #6c4079 100%);

  background: linear-gradient(to top right, #fc2c77 0%, #6c4079 100%);

}

.bg-gra-03 {

  background: -webkit-gradient(linear, left bottom, right top, from(#08aeea), to(#b721ff));

  background: -webkit-linear-gradient(bottom left, #08aeea 0%, #b721ff 100%);

  background: -moz-linear-gradient(bottom left, #08aeea 0%, #b721ff 100%);

  background: -o-linear-gradient(bottom left, #08aeea 0%, #b721ff 100%);

  background: linear-gradient(to top right, #08aeea 0%, #b721ff 100%);

}

/\* ==========================================================================

   #SPACING

   ========================================================================== \*/

.p-t-100 {

  padding-top: 100px;

}

.p-t-130 {

  padding-top: 130px;

}

.p-t-180 {

  padding-top: 180px;

}

.p-t-45 {

  padding-top: 45px;

}

.p-t-20 {

  padding-top: 20px;

}

.p-t-15 {

  padding-top: 15px;

}

.p-t-10 {

  padding-top: 10px;

}

.p-t-30 {

  padding-top: 30px;

}

.p-b-100 {

  padding-bottom: 100px;

}

.p-b-50 {

  padding-bottom: 50px;

}

.m-r-45 {

  margin-right: 45px;

}

.m-r-55 {

  margin-right: 55px;

}

.m-b-55 {

  margin-bottom: 55px;

}

/\* ==========================================================================

   #WRAPPER

   ========================================================================== \*/

.wrapper {

  margin: 0 auto;

}

.wrapper--w960 {

  max-width: 960px;

}

.wrapper--w790 {

  max-width: 790px;

}

.wrapper--w780 {

  max-width: 780px;

}

.wrapper--w680 {

  max-width: 680px;

}

/\* ==========================================================================

   #BUTTON

   ========================================================================== \*/

.btn {

  display: inline-block;

  line-height: 50px;

  padding: 0 50px;

  -webkit-transition: all 0.4s ease;

  -o-transition: all 0.4s ease;

  -moz-transition: all 0.4s ease;

  transition: all 0.4s ease;

  cursor: pointer;

  font-size: 15px;

  text-transform: uppercase;

  font-weight: 700;

  color: #fff;

  font-family: inherit;

}

.btn--radius {

  -webkit-border-radius: 3px;

  -moz-border-radius: 3px;

  border-radius: 3px;

}

.btn--radius-2 {

  -webkit-border-radius: 5px;

  -moz-border-radius: 5px;

  border-radius: 5px;

}

.btn--pill {

  -webkit-border-radius: 20px;

  -moz-border-radius: 20px;

  border-radius: 20px;

}

.btn--green {

  background: #57b846;

}

.btn--green:hover {

  background: #4dae3c;

}

.btn--blue {

  background: #4272d7;

}

.btn--blue:hover {

  background: #3868cd;

}

.btn--red {

  background: #ff4b5a;

}

.btn--red:hover {

  background: #eb3746;

}

/\* ==========================================================================

   #DATE PICKER

   ========================================================================== \*/

td.active {

  background-color: #2c6ed5;

}

input[type="date" i] {

  padding: 14px;

}

.table-condensed td, .table-condensed th {

  font-size: 14px;

  font-family: "Roboto", "Arial", "Helvetica Neue", sans-serif;

  font-weight: 400;

}

.daterangepicker td {

  width: 40px;

  height: 30px;

}

.daterangepicker {

  border: none;

  -webkit-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

  -moz-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

  box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

  display: none;

  border: 1px solid #e0e0e0;

  margin-top: 5px;

}

.daterangepicker::after, .daterangepicker::before {

  display: none;

}

.daterangepicker thead tr th {

  padding: 10px 0;

}

.daterangepicker .table-condensed th select {

  border: 1px solid #ccc;

  -webkit-border-radius: 3px;

  -moz-border-radius: 3px;

  border-radius: 3px;

  font-size: 14px;

  padding: 5px;

  outline: none;

}

/\* ==========================================================================

   #FORM

   ========================================================================== \*/

input {

  outline: none;

  margin: 0;

  border: none;

  -webkit-box-shadow: none;

  -moz-box-shadow: none;

  box-shadow: none;

  width: 100%;

  font-size: 14px;

  font-family: inherit;

}

.radio-container {

  display: inline-block;

  position: relative;

  padding-left: 30px;

  cursor: pointer;

  font-size: 16px;

  color: #666;

  -webkit-user-select: none;

  -moz-user-select: none;

  -ms-user-select: none;

  user-select: none;

}

.radio-container input {

  position: absolute;

  opacity: 0;

  cursor: pointer;

}

.radio-container input:checked ~ .checkmark {

  background-color: #e5e5e5;

}

.radio-container input:checked ~ .checkmark:after {

  display: block;

}

.radio-container .checkmark:after {

  top: 50%;

  left: 50%;

  -webkit-transform: translate(-50%, -50%);

  -moz-transform: translate(-50%, -50%);

  -ms-transform: translate(-50%, -50%);

  -o-transform: translate(-50%, -50%);

  transform: translate(-50%, -50%);

  width: 12px;

  height: 12px;

  -webkit-border-radius: 50%;

  -moz-border-radius: 50%;

  border-radius: 50%;

  background: #57b846;

}

.checkmark {

  position: absolute;

  top: 50%;

  -webkit-transform: translateY(-50%);

  -moz-transform: translateY(-50%);

  -ms-transform: translateY(-50%);

  -o-transform: translateY(-50%);

  transform: translateY(-50%);

  left: 0;

  height: 20px;

  width: 20px;

  background-color: #e5e5e5;

  -webkit-border-radius: 50%;

  -moz-border-radius: 50%;

  border-radius: 50%;

  -webkit-box-shadow: inset 0px 1px 3px 0px rgba(0, 0, 0, 0.08);

  -moz-box-shadow: inset 0px 1px 3px 0px rgba(0, 0, 0, 0.08);

  box-shadow: inset 0px 1px 3px 0px rgba(0, 0, 0, 0.08);

}

.checkmark:after {

  content: "";

  position: absolute;

  display: none;

}

.input--style-5 {

  background: #e5e5e5;

  line-height: 50px;

  -webkit-border-radius: 5px;

  -moz-border-radius: 5px;

  border-radius: 5px;

  padding: 0 22px;

  font-size: 16px;

  color: #555;

}

.input-group-desc {

  position: relative;

}

@media (max-width: 767px) {

  .input-group-desc {

    margin-bottom: 40px;

  }

}

.input-group {

  position: relative;

  margin: 0;

}

.label {

  color: #555;

  font-size: 15px;

  font-weight: 700;

}

.label--block {

  width: 100%;

}

.label--desc {

  position: absolute;

  text-transform: capitalize;

  display: block;

  color: #999;

  font-size: 14px;

  margin: 0;

  margin-top: 7px;

  left: 0;

}

/\* ==========================================================================

   #SELECT2

   ========================================================================== \*/

.select--no-search .select2-search {

  display: none !important;

}

.select2-container--open .select2-dropdown--below {

  border: none;

  -webkit-border-radius: 3px;

  -moz-border-radius: 3px;

  border-radius: 3px;

  -webkit-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

  -moz-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

  box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

  border: 1px solid #e0e0e0;

  margin-top: 5px;

  overflow: hidden;

}

.select2-container--default .select2-results\_\_option {

  padding-left: 22px;

}

.rs-select2 .select2-container {

  width: 100% !important;

  outline: none;

  background: #e5e5e5;

  -webkit-border-radius: 5px;

  -moz-border-radius: 5px;

  border-radius: 5px;

}

.rs-select2 .select2-container .select2-selection--single {

  outline: none;

  border: none;

  height: 50px;

  background: transparent;

}

.rs-select2 .select2-container .select2-selection--single .select2-selection\_\_rendered {

  line-height: 50px;

  padding-left: 0;

  color: #555;

  font-size: 16px;

  font-family: inherit;

  padding-left: 22px;

  padding-right: 50px;

}

.rs-select2 .select2-container .select2-selection--single .select2-selection\_\_arrow {

  height: 50px;

  right: 15px;

  display: -webkit-box;

  display: -webkit-flex;

  display: -moz-box;

  display: -ms-flexbox;

  display: flex;

  -webkit-box-pack: center;

  -webkit-justify-content: center;

  -moz-box-pack: center;

  -ms-flex-pack: center;

  justify-content: center;

  -webkit-box-align: center;

  -webkit-align-items: center;

  -moz-box-align: center;

  -ms-flex-align: center;

  align-items: center;

}

.rs-select2 .select2-container .select2-selection--single .select2-selection\_\_arrow b {

  display: none;

}

.rs-select2 .select2-container .select2-selection--single .select2-selection\_\_arrow:after {

  font-family: "Material-Design-Iconic-Font";

  content: '\f2f9';

  font-size: 24px;

  color: #999;

  -webkit-transition: all 0.4s ease;

  -o-transition: all 0.4s ease;

  -moz-transition: all 0.4s ease;

  transition: all 0.4s ease;

}

.rs-select2 .select2-container.select2-container--open .select2-selection--single .select2-selection\_\_arrow::after {

  -webkit-transform: rotate(-180deg);

  -moz-transform: rotate(-180deg);

  -ms-transform: rotate(-180deg);

  -o-transform: rotate(-180deg);

  transform: rotate(-180deg);

}

/\* ==========================================================================

   #TITLE

   ========================================================================== \*/

.title {

  font-size: 24px;

  text-transform: uppercase;

  font-weight: 700;

  text-align: center;

  color: #fff;

}

/\* ==========================================================================

   #CARD

   ========================================================================== \*/

.card {

  -webkit-border-radius: 3px;

  -moz-border-radius: 3px;

  border-radius: 3px;

  background: #fff;

}

.card-5 {

  background: #fff;

  -webkit-border-radius: 10px;

  -moz-border-radius: 10px;

  border-radius: 10px;

  -webkit-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

  -moz-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

  box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

}

.card-5 .card-heading {

  padding: 20px 0;

  background: #1a1a1a;

  -webkit-border-top-left-radius: 10px;

  -moz-border-radius-topleft: 10px;

  border-top-left-radius: 10px;

  -webkit-border-top-right-radius: 10px;

  -moz-border-radius-topright: 10px;

  border-top-right-radius: 10px;

}

.card-5 .card-body {

  padding: 52px 85px;

  padding-bottom: 73px;

}

@media (max-width: 767px) {

  .card-5 .card-body {

    padding: 40px 30px;

    padding-bottom: 50px;

  }

}

**Index.html**

{% load static %}

<!DOCTYPE html>

<head>

    <style>

        .xyz{

          margin-bottom: 0px;

          background-image: url('{% static "image/homepage.jpg" %}');

          background-size: cover;

          background-repeat: no-repeat;

        }

      </style>

</head>

<body>

  {% include "blood/navbar.html" %}

<br>

<section id="section-jumbotron" style="margin-bottom: 0px;" class="jumbotron jumbotron-fluid d-flex justify-content-center align-items-center xyz">

    <div class="container text-center">

        <br><br><br><br><br><br><br><br><br><br><br><br><br><br>

      <br><br><br><br>

    </div>

  </section>

  <div class="jumbotron" style="margin-top: 0px;margin-bottom: 0px;">

    <p class="lead text-center">“YOU DON'T NEED TO BE A DOCTOR TO BE A HERO, JUST DONATE YOUR BLOOD.”

      </p>

  </div>

  {% include "blood/footer.html" %}

</body>

</html>

<!--

Developed By : Dhawan Bhartiya, Snehal Rani and Ankit Kumar

-->

**CONCLUSION AND FUTURE SCOPE**

Based on results, this study concluded that Blood bank management system is much better than the manual system. The findings showed that respondents prefer to use online blood bank management system rather than the manual system because it offers many advantages and benefits that lead to its effectiveness, and efficiency. Because of the increased confidence on the users on the system, it can be concluded that the Blood bank management system enhances blood transfusion safety because it provides better ways of handling the various processes in blood bank. The blood management system has been experimentally proven to work satisfactorily by developing web applications and the system can be by donor and user etc.

Future Scope:

a. In future we will update our system continuously

b. We will develop live chat feature

c. Track location of the donor etc