VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Jnana Sangama, Belagavi-590010

DBMS MINI PROJECT REPORT

ON

"BLOOD BANK MANAGEMENT SYSTEM"

Submitted in partial fulfillment for the requirements for the fifth semester

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

For the Academic Year 2020-2021

Submitted by:

SUJITH N.E	**1MV18CS113**
TEJA E	**1MV18CS117**
Y SAI GOKUL	**1MV18CS127**

Under the guidance of:

Dr. Suma Swamy

Professor, Department of CSE

Sir M. Visvesvaraya Institute of Technology, Bengaluru

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY

HUNASAMARANAHALLI, BENGALURU-562157

SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY
Krishnadevaraya Nagar, International Airport Road,
Hunasmaranahalli, Bengaluru – 562157
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

It is certified that the **DBMS Mini Project work** entitled " **BLOOD BANK MANAGEMENT SYSTEM**" is carried out by **SUJITH N.E (1MV18CS113), TEJA E (1MV18CS117), Y SAI GOKUL (1MV18CS127)** bonafide students of **Sir M Visvesvaraya Institute of Technology** in partial fulfillment for the 5th semester for the award of the Degree of Bachelor of Engineering in Computer Science and Engineering of the **Visvesvaraya Technological University, Belagavi** during the academic year **2020-2021**.It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the course of Bachelor of Engineering.

Name & Signature of Guide	Name & Signature of HOD	Name & Signature of Principal
Dr . Suma Swamy Prof & Internal Guide Dept. Of CSE, Sir MVIT Bengaluru - 562157	**Dr. G. C. Bhanu Prakash** HOD, Dept. Of CSE, Sir MVIT Bengaluru - 562157	**Dr. V.R. Manjunath** Principal, Sir MVIT Bengaluru – 562157

External Examination:

Name of Examiner	Signature with Date
1)	
2)	

DECLARATION

We hereby declare that the entire project work embodied in this dissertation has been carried out by us and no part has been submitted for any degree or diploma of any institution previously.

Place: Bengaluru

Date:

Signature of Students:

SUJITH N.E
(1MV18CS113)

TEJA E
(1MV18CS117)

Y SAI GOKUL
(1MV1CS127)

ACKNOWLEDGMENT

It gives us immense pleasure to express our sincere gratitude to the management of **Sir M. Visvesvaraya Institute of Technology,** Bengaluru for providing the opportunity and the resources to accomplish our project work in their premises.

On the path of learning, the presence of an experienced guide is indispensable and we would like to thank our guide **Dr Suma Swamy,** Professor, Dept. of CSE, for her invaluable help and guidance.

Heartfelt and sincere thanks to **Dr. G. C. Bhanu Prakash,** HOD, Dept. of CSE, for his suggestions, constant support and encouragement.

We would also like to convey our regards to **Dr. V.R. Manjunath,** Principal, Sir. MVIT for providing us with the infrastructure and facilities needed to develop our project.

We would also like to thank the staff of Department of Computer Science and Engineering and lab-in-charges for their co-operation and suggestions. Finally, we would like to thank all our friends for their help and suggestions without which completing this project would not have been possible.

- SUJITH N E	1MV18CS113
- TEJA E	1MV18CS117
- Y SAI GOKUL	1MV18CS127

ABSTRACT

The major concern for hospitals and blood centres today is the wastage of blood products and blood transfusion errors. Usually there is a severe shortage in our country between blood requirement and blood availability, as a result, many patients die or suffer unnecessarily because they have no access to blood and blood products. In order to resolve this problem, we have created the project blood bank and donor system which is a web project. The main aim of this project will therefore be to find more effective ways of managing the database of blood banks and blood donors and establish a forum for people connected to potential blood donors in the region to help get blood and blood products easier and faster while not wasting the blood by letting them outlive their shelf life.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Blood is a necessary element in the human body. Without blood, the human body is incomplete. Blood consists about 7% to 8% of human weight, and is essential for the body to function. Usually there is a severe shortage between blood requirement and blood availability in a populous country, as a result, many patients die or suffer unnecessarily because they have no access to blood and blood products. In order to resolve this problem, we have to make sure that there is a required amount of blood in any particular region such that there is no less than the required amount and not too much more than the needed so that wastage of blood is minimized. Here our objective is to bridge the gap between the blood requirement and blood reserves. Blood has a expiry date of around 40 days, a lot of blood and blood products go unused and expire resulting in a shortage of blood. those who donated cannot donate blood for another 90 days resulting in a issues such an unavailability of blood during the needed time and wastage during times when not needed hence we hope to create a portal which records the contact information of available donors during a certain time period and aim to minimize the wastage of blood by making it convenient to contact donors during time of need . our project records details of donors and uses the real time database to generate list of available donors at a given time and their contact information . we also help minimize wastage of blood by using the system. Because of enormous amount of donor data there must be an efficient and successful way of managing data that could make the online blood donation site a pavestone .

The main aim of this project will therefore be to find more effective ways of managing the database of blood banks and blood donors and establish a forum for people to be connected to potential blood donors in the region to help get blood and blood products easier and faster while not wasting the blood by letting them outlive their shelf life.

1.2 Need for BBMS

Blood transfusion is an essential component of every country's health care system and patients needing blood transfusion. there is severe shortage in our country between blood requirement and blood reserves , and as a result , many patients die or suffer unnecessarily because they have a no access to blood and blood products in order to resolve this problem , we have created the project blood bank and donor system which is a web project and can be easily connect to anything via the internet service hence online platform therefore the best choice for our project

1.3 Aim/ Objective of this Project

Here our main aim and objective is to bridge the gap between the blood requirement and blood reserves . each year 1.3 crore units of blood are needed out only 90 lakh units are collected is wasted many times due to the expiry date of blood which is around 35 days a lot of blood and blood products go unused and expire resulting in a shortage of blood . those who donated cannot donate blood for another 90 days resulting in a issues such an unavailability of blood during the needed time and wastage during times when not needed hence we hope to create a portal which records the contact information of available donors during a certain time period and aim to minimize the wastage of blood by making it convenient to contact donors during time of need .

Our project records details of donors and uses the database to generate list of available donors at a given time and their contact information . we also help minimize wastage of blood by using the system.

Because of enormous amount of donor data there must be an efficient and successful way of managing data that could make the online blood donation site a pavestone . the main aim of this project will therefore be to find more effective ways of managing database of blood banks and blood donors and establish a forum for people connected to potential blood donors of the region

CHAPTER 2

FRONT END AND BACK END

2.1 PYTHON

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

2.2 SQLITE3

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database file format is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between big-endian and little-endian architectures. These features make SQLite a popular choice as an Application File Format

2.3 DJANGO

Django's primary goal is to ease the creation of complex, database-driven websites . It uses the principles of less code, low coupling, rapid development and the principle of don’t repeat yourself which means developers can do more than one iteration at a time without starting the whole work schedule from scratch.

2.4 HTML

Hyper Text Markup Language (HTML) is the main markup language for creating web pages and other information that can be displayed in a web browser. HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>), within in the web page content. HTML tags most commonly come in pairs like <h1> and </ h1>, although some tags, known as empty elements, are unpaired, for example <img>. The first tag in a pair is the start tag, the second tag is the end tag (they are also called opening tag and closing tag). In between these tags web designers can add text, tags, comments and other types of text-based content. The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages. Web browser can also refer to Cascading Style Sheet (CSS) to define the appearance and layout of text and other material. The W3C, maintainer of both the HTML and the CSS over explicit presentational HTML markup.

2.5 CSS

Cascading style Sheets (CSS). Is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup-language. It's most common application is to style web pages written in HTML and XHTML, but the language can also. Be applied to any kind of XML document, including main XML, SVG and XUL. CSS is designed primarily to enable the separation of document content (written in HTML or similar markup language) from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content (such as by allowing for table less web design) CSS can also allow the same markup page to be presented in different styles for different rendering methods, such as on-screen, in print, by voice (when read out by a speech- based browser or screen reader) and on Braille-based, tactile devices. It can also be used to allow the web page to display differently depending on the screen size or device on which it is being viewed. While the author of a document typically links that document to a CSS style sheet, readers can use a different style sheet, perhaps one on their. Own computer, to override the one the author has specified.

2.6 JAVASCRIPT

JAVA SCRIPT (often shortened to **JS**) is a lightweight, interpreted, object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well. It is a prototype-based, multi-paradigm scripting language that is dynamic, and supports object-oriented, imperative, and functional programming styles.JavaScript runs on the client side of the web, which can be used to design / program how the web pages behave on the occurrence of an event. JavaScript is an easy to learn and also powerful scripting language, widely used for controlling web page behavior.JavaScript can function as both a procedural and an object oriented language. Objects are created programmatically in JavaScript, by attaching methods and properties to otherwise empty objects **at run time**, as opposed to the syntactic class definitions common in compiled languages like C++ and Java. Once an object has been constructed it can be used as a blueprint (or prototype) for creating similar objects.JavaScript's dynamic capabilities include runtime object construction, variable parameter lists, function variables, dynamic script creation (via eval), object introspection (via for ... in), and source code recovery (JavaScript programs can decompile function bodies back into their source text).

CHAPTER 3

SPECIFICATIONS

3.1 HARDWARE REQUIREMENTS

· Processor: Pentium-III (or) Higher

· Ram: 256MB (or) Higher

· Hard disk: 2GB (or) Higher

3.2 SOFTWARE REQUIREMENTS

- Technology: Python Django
- IDE : Pycharm/Atom
- Client Side Technologies: HTML, CSS, JavaScript , Bootstrap
- Server Side Technologies: Python
- Data Base Server: Sqlite
- Operating System: Microsoft Windows/Linux

3.3 FUNCTIONAL REQUIREMENTS

· Administrative functions – The web app has to have administrative functions to keep the app running

· Authorization levels – The app needs a level of authorization in hierarchical system to help manage work more efficiently

· Reporting Requirements – There need to be some requirements such that the app can report emergency needs of a patient to an admin

· Historical data – Data through the years has to be recorded and stored to create a more comprehensive database

· Legal and Regulatory Requirements – The legal and regulatory system of the government must be met

3.4 NON-FUNCTIONAL REQUIREMENTS

- Maintainability – The app and the database should be easy to maintain and function
- Manageability – The app and database needs to be managed properly hence a proper database system is needed
- Environmental – The app and the database should be appealing in this competitive environment
- Data integrity – The data in the database should be secure and easy to access while maintaining confidentiality of users
- Scalability – The app and database should be easy to scale and update to meet the requirements around the globe easily

CHAPTER 4

SYSTEM ANALYSIS

4.1 EXISTING SYSTEM

The current system is not efficient in managing blood . there is either deficiency or excess amount of blood in most cases . now a days there are frequent blood camps being organised by multiple social and corporate organisations on different occasions . many times the number of donors donating is more than the actual requirement because of which excess amount of blood is being collected blood can't be stored very long and donors cannot donate blood blood for the next 90 days . Situations like blood and donors not available at the same time are arising because of this process.

4.2 PROPOSED SYSTEM

A realtime database system that records blood transactions and makes use of preexisting data to predict the amount of blood needed for a given period time in a particular area with low divergence consistently and records the user donations and creates a forum for donors and recipients to interact.

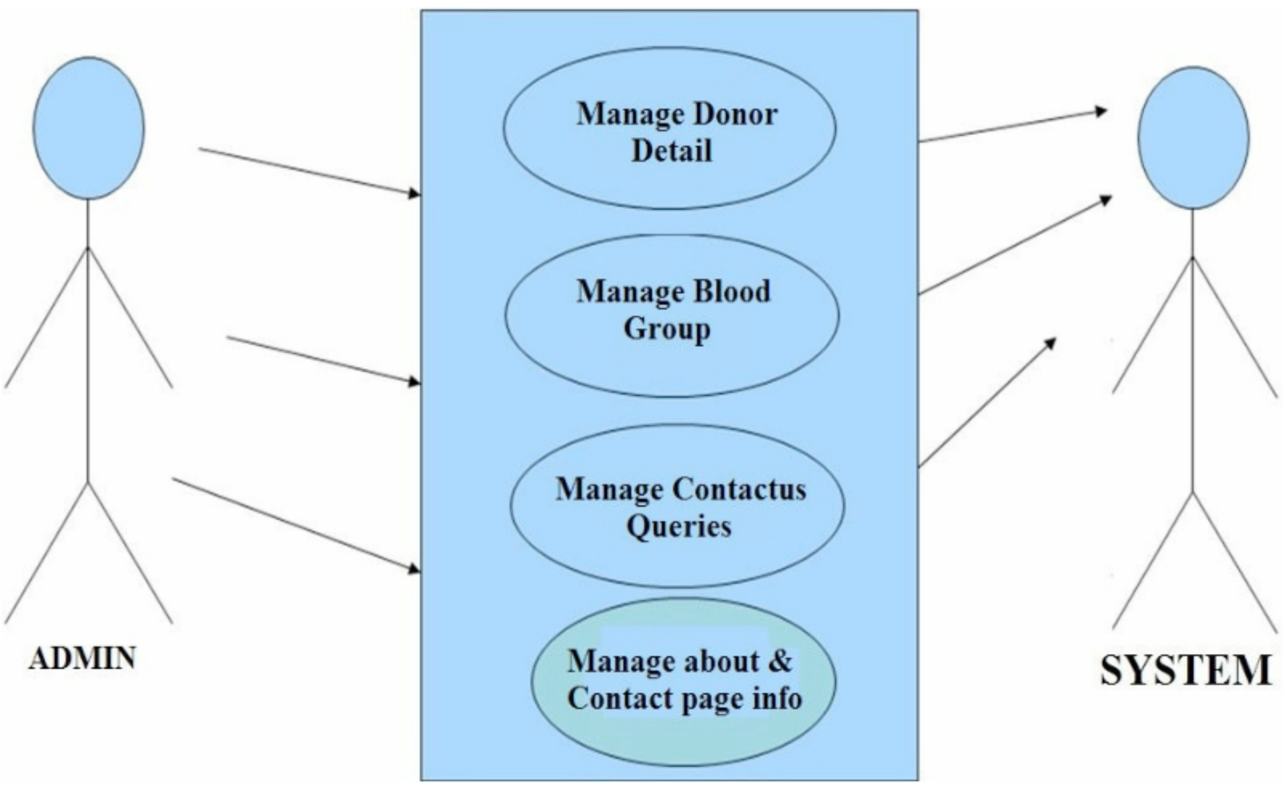
CHAPTER 5

SYSTEM DESIGN

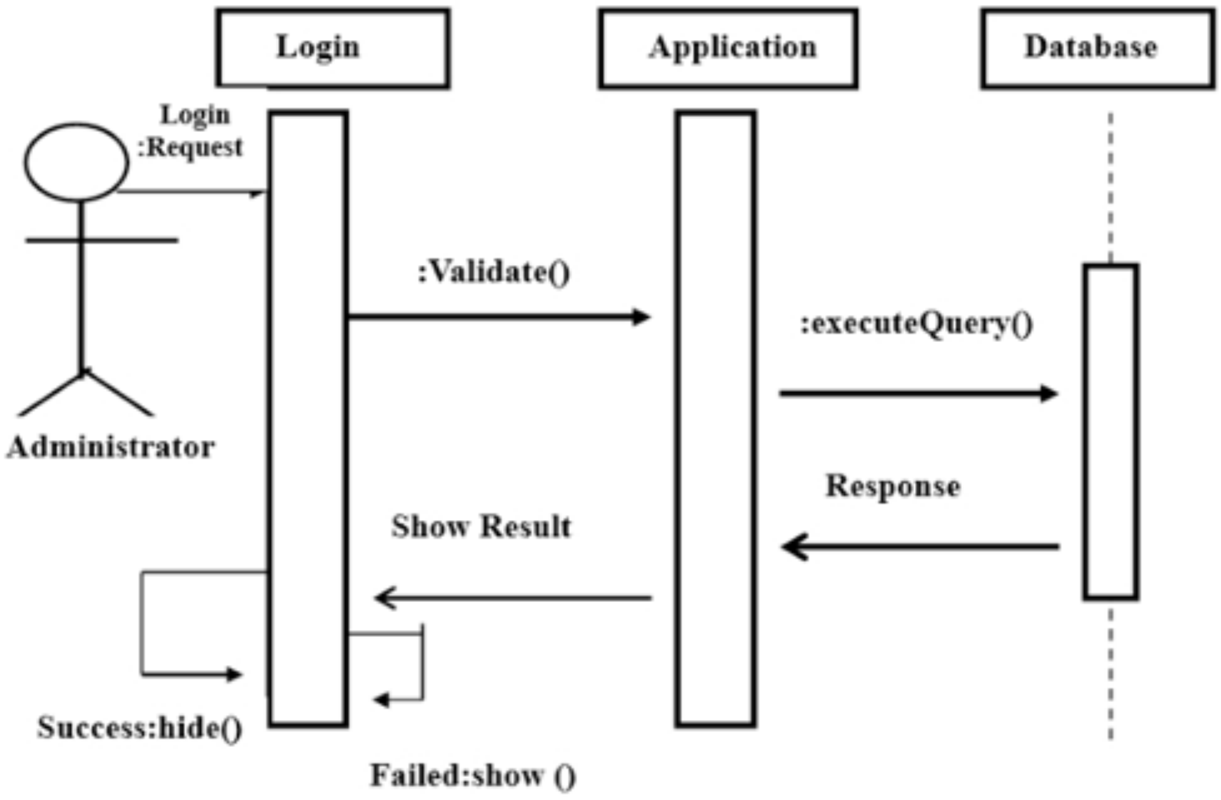
5.1 USE CASE DIAGRAM

Use case diagram consists of use cases and actors and shows the interaction between them. The key points are:

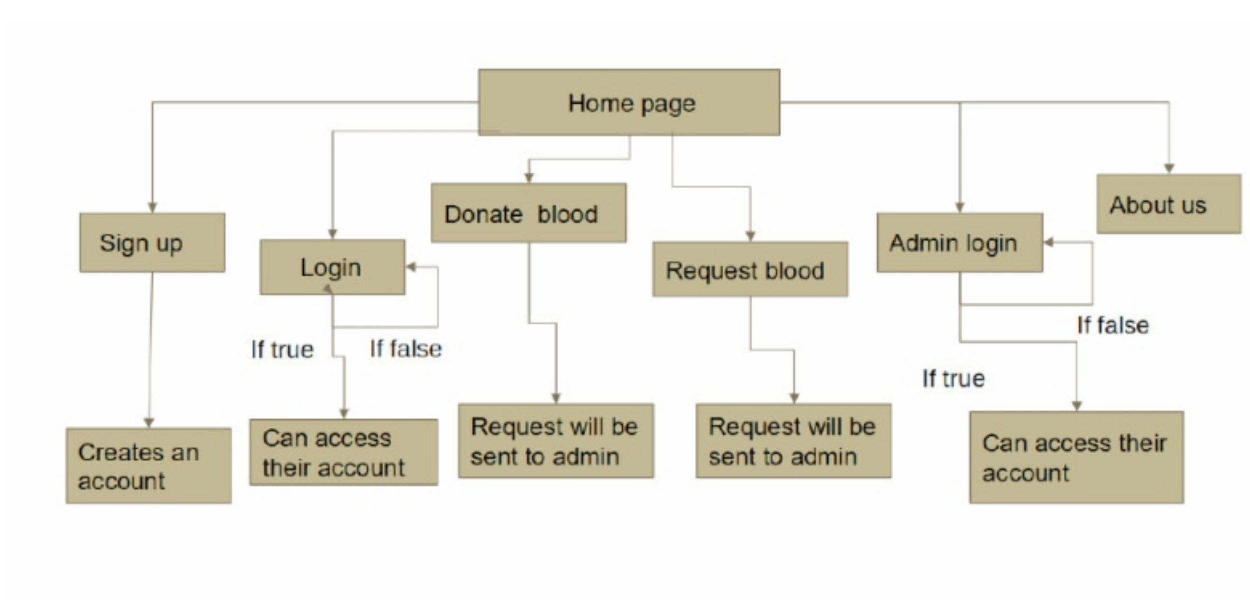
- The main purpose is to show the interaction between the use cases and the actor.
- To represent the system requirement from user's perspective.
- The use cases are the functions that are to be performed in the module.



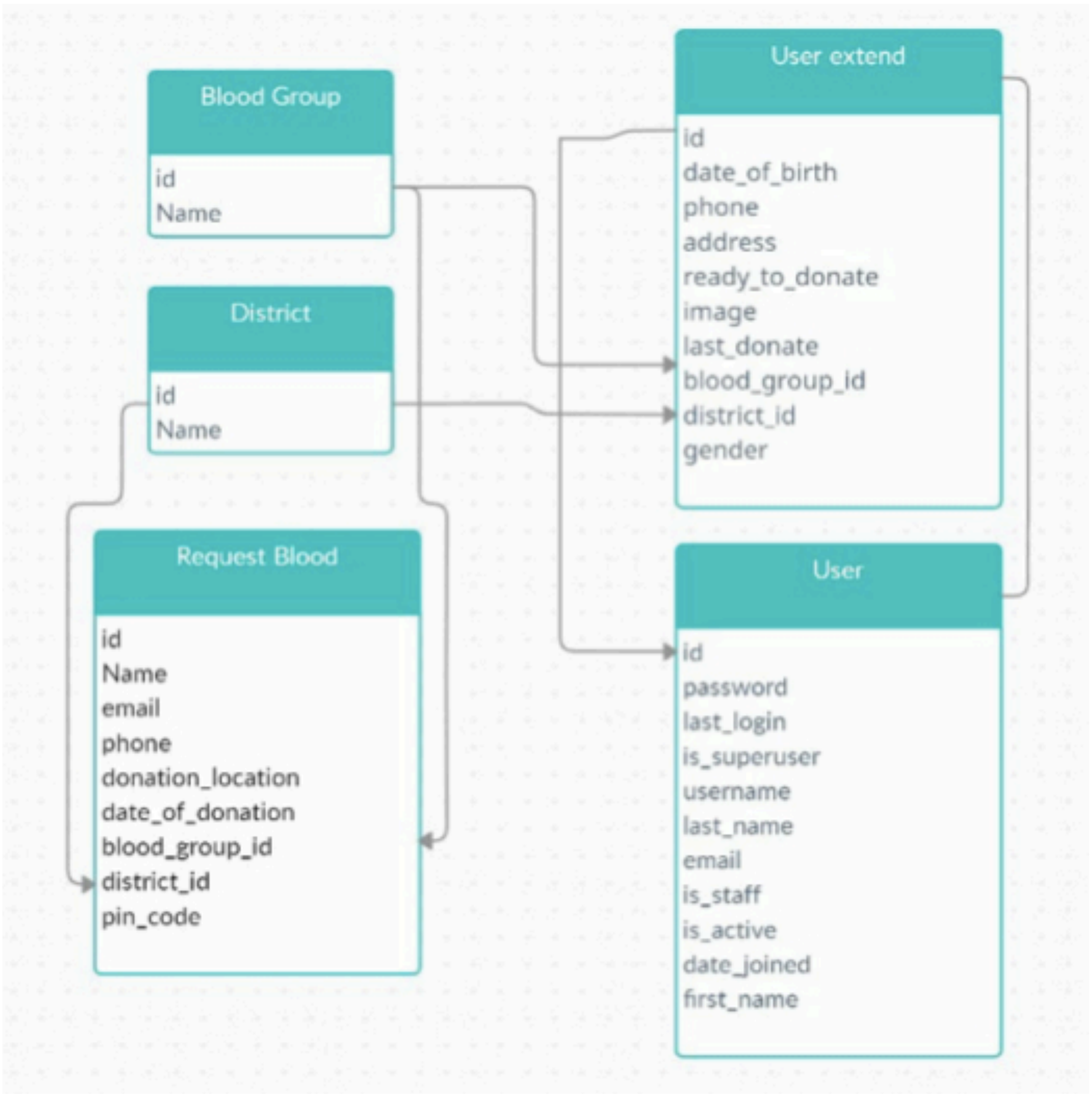
5.2 SEQUENCE DIAGRAM



5.3 DATA FLOW DIAGRAM



5.4 ER DIAGRAM



The diagram above represents the entity relationship diagram of our project The donor gives all their details to register in our database and our staff process their data along with the admins they manage the inventory of our blood bank . When a patient needs blood if it is not available in the inventory then our team requests the donors registered with us if they can donate blood and arrange for the necessary things to do We also record the recipient data for further reference and usage in the future.

CHAPTER 6

FRONTEND IMPLEMENTATION

6.1 Frontend Web pages

Creation of webpages for the website using frontend tools such as html,css, bootstrap and JavaScript.

Some of the webpages in the website are Home screen, Donate screen, Request screen etc. These webpages are designed in such a way that it is user friendly for the user. Modern UI and UX concepts are used while designing the frontend which makes the website to display the content in the most simpler and efficient manner.

6.2 Database Creation:

We have created Database using Sqlite3.
Our website database consists of 5 relations/ tables.

(i) Bloodgroup: This table consists information about the blood groups . The attributes of the table are:

· id – contains blood group id

· name – contains name of blood group

(ii) User : This table consists information about the users . The attributes of the table are:

· id – contains user id

· password– contains user password

· last_login – contains the last login time of the user

· is_superuser – shows if user is a superuser

· username – shows the username of the user

· first_name – shows the first name of the user

· last_name – shows the lastname of the user

· email – shows email address of the user

· is_staff – shows if user is a staff of our organization

· is_active – shows if user is active currently

· date_joined -Shows the date user joined our bbms (iii)Userextended : This table consists of complete information about the user. The attributes of the table are:

- id – contains user id
- DOB – contains date of birth
- phone – contains the phone number of the user
- address – contains the address of the user
- ready_to_donate – this button shows if user is ready to donate
- image – a image of the user is placed here
- last_donated – this records the last donation date
- blood_group_id – this shows the id of the blood group
- district_id – this shows the district id
- donor_id – this shows the donor id
- gender – shows the gender of the user

(iv) request blood : This table consists information about blood requests. The attributes of the table are:

- id – contains recipient id
- name – name of recipient
- phone – contains the phone number of the recipient
- email – contains the email address of the recipient
- donation_location – this records the location of the recipient
- date_of_donation – this records the last donation date
- blood_group_id – this shows the id of the blood group
- district_id – this shows the district id
- pin_code – this shows the pin code of the city in which the recipient resides

(v) district : This table consists information about the districts . The attributes of the table are:

- id – contains district id
- name – name of the district

6.3 Authentication System:

User can login through a Username and Password:

Users are asked to sign up in our webpage where they fill their details to create a username and password which can be used for future use

6.4 Backend System:

We have developed the backend of the website using Django.We link the frontend and the sqlite3 database .The Django Client sends requests to the backend and handles these requests by calling the appropriate routes. The requests from the client side communicates Django client which connects to the sqlite3 database and gets the required data and sends this data to the frontend client in the form of data.

For Example when a user wants to create an account in the website, the Django client sends the user details to the server side, the server side creates as new user in the database if the user is not yet created an account is created else it sends information to the Django client saying the user has already has a account.

In the same way various functionalities such as requesting and donating blood are handled by the server side.

6.5 Deployment of the website

Currently we are deploying our website in a local network and it can be accessed only when we deploy and run the server .

CHAPTER 7

7. SOURCE CODE

The Source Code of the Project is uploaded in google drive .

The link of the repository: https://drive.google.com/file/d/1zuIDBb5s4BYcH5AjXvH5wc984pMZhRM8/view?usp=sharing

- MODELS.PY

```
from django.db import models
from django.contrib.auth.models import User

GENDER = [
    ('Male','Male'),
    ('Female','Female'),
    ('Other','Other')
]

class BloodGroup(models.Model):
    name = models.CharField(max_length=5)
    def __str__(self):
        return self.name

class District(models.Model):
    name = models.CharField(max_length=100)

    def __str__(self):
        return self.name

class UserExtend(models.Model):
    donor = models.OneToOneField(User, on_delete=models.CASCADE)
    date_of_birth = models.DateField(help_text="yyyy-mm-dd")
    phone = models.CharField(max_length=30, unique=True)
    address = models.CharField(max_length=100)
```

```
district = models.ForeignKey(District,
on_delete=models.CASCADE)
    blood_group = models.ForeignKey(BloodGroup,
on_delete=models.CASCADE)
    gender = models.CharField(choices=GENDER, max_length=10)
    ready_to_donate = models.BooleanField(default=True)
    image = models.ImageField(verbose_name="Profile Picture",
upload_to="images/")
    last_donate = models.DateField(help_text="yyyy-mm-dd")

    def __str__(self):
        return self.donor.username


class RequestBlood(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField()
    phone = models.CharField(max_length=20)
    donation_location = models.CharField(max_length=100)
    district = models.ForeignKey(District,
on_delete=models.CASCADE)
    blood_group = models.ForeignKey(BloodGroup,
on_delete=models.CASCADE)
    date_of_donation = models.DateField(help_text="yyyy-mm-
dd")
    pin_code = models.IntegerField(help_text='You can edit
your request later using this unique code', unique=True)

    def __str__(self):
        return self.name
```

VIEWS.PY

```
from django.shortcuts import render, get_object_or_404, redirect
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout,
update_session_auth_hash
from django.contrib.auth.decorators import login_required
from .models import UserExtend,RequestBlood,District,BloodGroup
from .forms import UserForm1, UserForm2, LoginForm, RequestForm,
ChangeForm1, ChangeForm2
from django.db.models import Count
from django.contrib.auth.forms import PasswordChangeForm


def homepage(request):
    return render(request, "blood_app/home.html")
def adminbase(request):
    return render(request, "blood_app/adminbase.html")


def registerView(request):
    if request.method == "POST":
        form1 = UserForm1(request.POST)
        form2 = UserForm2(request.POST, request.FILES)
        if form1.is_valid() and form2.is_valid():
            obj = form1.save(commit=False)
            obj.set_password(obj.password)
            obj.save()
            obj2 = form2.save(commit=False)
            obj2.donor = obj
            obj2.save()
            return redirect('login')
    else:
        form1 = UserForm1()
        form2 = UserForm2()

    return render(request, "blood_app/register.html",
{‘form1':form1,'form2':form2})
```

```
def loginView(request):
    if request.method == "POST":
        form = LoginForm(request.POST)
        if form.is_valid():
            cd = form.cleaned_data
            user = authenticate(request, username=cd['username'], 
password=cd['password'])
            if user is not None:
                login(request, user)
                return redirect('profile')
    else:
        form = LoginForm()
    return render(request, "blood_app/login.html", {'form':form})


def logoutView(request):
    logout(request)
    return redirect('home')

@login_required
def profileView(request):
    return render(request, "blood_app/profile.html")

def adminlogin(request):
    return render(request, "blood_app/adminlogin.html")

def adminhome(request):
    all_group = 
BloodGroup.objects.annotate(total=Count('userextend'))
    return render(request, "blood_app/adminlogged.html", 
{"all_group": all_group})
```

```
def adminlogged(request):
    ap = request.GET.get('adminpassword', 'default')
    au= request.GET.get('adminuserid', 'default')
    aus ="admin"
    aps= "admin"
    if ap==aps and au == aus :
        all_group =
BloodGroup.objects.annotate(total=Count('userextend'))
        return render(request, "blood_app/adminlogged.html",
{"all_group": all_group})
      else:
        all_group =
BloodGroup.objects.annotate(total=Count('userextend'))
        return  render ( request ,'blood_app/adminlogin.html')

def createReqView(request):
    if request.method == "POST":
        form = RequestForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('createreq')
    else:
        form = RequestForm()

    return render(request, "blood_app/brequest.html", {'form':form})

def allReqView(request):
    all_req = RequestBlood.objects.all()
    return render(request, "blood_app/allrequest.html",
{'all_req':all_req})

def groupView(request, id):
    obj = get_object_or_404(BloodGroup, pk=id)
    donor = UserExtend.objects.filter(blood_group=obj)
    return render(request, "blood_app/donorlist.html",
{'donor':donor})
```

```
def detailView(request, id):
    obj = get_object_or_404(User, pk=id)
    return render(request, "blood_app/details.html", {'obj':obj})

@login_required
def changePasswordView(request):
    if request.method == "POST":
        form = PasswordChangeForm(user=request.user, 
data=request.POST)
        if form.is_valid():
            form.save()
            update_session_auth_hash(request,form.user)
            return redirect('profile')
    else:
        form = PasswordChangeForm(user=request.user)

    return render(request, "blood_app/password.html", {'form':form})

@login_required
def editProfileView(request):
    if request.method == "POST":
        form1 = ChangeForm1(request.POST, instance=request.user)
        form2 = ChangeForm2(request.POST,request.FILES, 
instance=request.user.userextend)
        if form1.is_valid() and form2.is_valid():
            obj = form1.save()
            obj2 = form2.save(commit=False)
            obj2.donor = obj
            obj2.save()
            return redirect('profile')
    else:
        form1 = ChangeForm1(instance=request.user)
        form2 = ChangeForm2(instance=request.user.userextend)
    return render(request, "blood_app/edit_profile.html", 
{'form1':form1,'form2':form2})
```

```
@login_required
def statusView(request):
    obj = request.user.userextend
    if obj.ready_to_donate:
        obj.ready_to_donate = False
        obj.save()
    else:
        obj.ready_to_donate = True
        obj.save()
    return redirect('profile')


def getRequestView(request):
    if request.method == "POST":
        pin = request.POST.get("pin")
        obj = get_object_or_404(RequestBlood, pin_code=int(pin))
        return render(request, "blood_app/get_request.html",
{'obj':obj})
     return render(request, "blood_app/get_request.html")

def editRequestView(request, pin):
    obj = get_object_or_404(RequestBlood, pin_code=pin)
    if request.method == "POST":
        form = RequestForm(request.POST, instance=obj)
        if form.is_valid():
            form.save()
            return redirect('home')
    else:
        form = RequestForm(instance=obj)

    return render(request, "blood_app/edit_request.html",
{'form':form})

def deleteRequestView(request, pin):
    obj = get_object_or_404(RequestBlood, pin_code=pin)
    obj.delete()
    return redirect('home')
```

FORMS.PY

```
from django import forms
from django.contrib.auth.models import User
from .models import UserExtend,RequestBlood,District
class UserForm1(forms.ModelForm):
    class Meta:
        model = User
        fields = 
['username','first_name','last_name','email','password']
        widgets = {
            'password': forms.PasswordInput,
        }
class UserForm2(forms.ModelForm):
    class Meta:
        model = UserExtend
        exclude = ('donor','ready_to_donate')

class LoginForm(forms.Form):
    username = forms.CharField()
    password = forms.CharField(widget=forms.PasswordInput())

class RequestForm(forms.ModelForm):
    class Meta:
        model = RequestBlood
        fields = "__all__"


class ChangeForm1(forms.ModelForm):
    class Meta:
        model = User
        fields = ['username','first_name','last_name','email']


class ChangeForm2(forms.ModelForm):
    class Meta:
        model = UserExtend
        exclude = (‘donor','ready_to_donate')
```

URLS.PY

```
from django.contrib import admin
from django.urls import path
from blood_app import views
from django.conf import settings
from django.conf.urls.static import static
from django.contrib.staticfiles.urls import staticfiles_urlpatterns

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.homepage, name='home'),
    path('register/', views.registerView, name='register'),
    path('login/', views.loginView, name='login'),
    path('logout/', views.logoutView, name='logout'),
    path('adminlogin/', views.adminlogin,name = 'adminlogin'),
    path('adminlogin/adminlogged/', views.adminhome,name =
'adminlogged'),
path('adminbase', views.adminbase,name= 'adminbase'),
    path('profile/', views.profileView, name='profile'),
    path('profile/changepassword/', views.changePasswordView,
name='changepassword'),
    path('profile/update/', views.editProfileView,
name='updateprofile'),
    path('profile/status/', views.statusView, name='status'),
    path('request/homepage/', views.homepage, name='homepages'),
    path('request/', views.createReqView, name=“createreq"),
path('editrequest/<int:pin>/', views.editRequestView,
name="editrequest"),
    path('deleterequest/<int:pin>/', views.deleteRequestView,
name="deleterequest"),
    path('getrequest/', views.getRequestView, name="getrequest"),
    path('allrequest', views.allReqView, name='allrequest'),
    path('group/<int:id>/', views.groupView, name='group'),
    path('donor/<int:id>/', views.detailView, name="details"),

] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
urlpatterns+= staticfiles_urlpatterns()
```

SETTINGS.PY

```
import os

# Build paths inside the project like this:
os.path.join(BASE_DIR, ...)
BASE_DIR =
os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/
checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '1irrbvga=w-aqyd$edc$fz6t@_sxkj(yyp2*&)tbw_oa6dtc03'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
ALLOWED_HOSTS = []
# Application definition
INSTALLED_APPS = [
    'blood_app',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

```
ROOT_URLCONF = 'bloodbank.urls'

TEMPLATES = [
    {
        'BACKEND':
'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',

'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'bloodbank.wsgi.application'


# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

```
# Password validation
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityVal
idator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]
# Internationalization
# https://docs.djangoproject.com/en/2.2/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/
STATIC_URL = '/static/'
STATICFILES_URL = [
    os.path.join(BASE_DIR,'static')
]
STATIC_ROOT = os.path.join(BASE_DIR,'static')
MEDIA_ROOT = os.path.join(BASE_DIR,"media")
MEDIA_URL = “/media/"
LOGIN_URL = '/login/'
```

HTML PAGES

->BASE.HTML

```
<!doctype html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">

  <link rel="stylesheet" href="https://use.fontawesome.com/
releases/v5.7.0/css/all.css"
  integrity="sha384-
lZN37f5QGtY3VHgisS14W3ExzMWZxybE1SJSEsQp9S+oqd12jhcu+A56Ebc1zFSJ"
  crossorigin="anonymous">
  <!-- Bootstrap CSS
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
bootstrap/4.5.0/css/bootstrap.min.css"
integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxF
fc+NcPb1dKGj7Sk" crossorigin="anonymous">-->

  <title>Blood Bank</title>
</head>

  <div class="header">
    <div class="container">
        <div >
            <a><h1 style = "color : white ; font-weight :
bold">Blood Bank Management System</h1></a>
        </div>
        <div class="top-menu">
```

```
{% if request.user.is_authenticated %}
            <ul>
            <li><a href="{% url 'profile' %}"><span class="fas fa-
home" aria-hidden="true"></span> Profile</a></li>

            <li><a href="{% url 'updateprofile' %}"><span
class="far fa-user" aria-hidden="true"></span> Update Profile</a></
li>
            <li><a href="{% url 'changepassword' %}"><span
class="far fa-calendar-alt" aria-hidden="true"></span> Change
Password</a></li>
            <li><a href="{% url 'logout' %}"><span class="fas fa-
sign-out-alt" aria-hidden="true"></span> Logout</a></li>
          </ul>
            {% else %}
            <ul>
              <li><a href="{% url 'home' %}"><span class="fas fa-
home" aria-hidden="true"></span> Home</a></li>
              <li><a href="{% url 'createreq' %}"><span class="fas
fa-user" aria-hidden="true"></span> Request For Blood</a></li>
              <li><a href="{% url 'getrequest' %}"><span class="far
fa-user" aria-hidden="true"></span> Edit Request</a></li>
              <li><a href="{% url 'register' %}"><span class="far
fa-calendar-alt" aria-hidden="true"></span> Become A Donar</a></li>
              <li><a href="{% url 'login' %}"><span class="fas fa-
sign-in-alt" aria-hidden="true"></span> Donor Login</a></li>
              <li><a href="{% url 'adminlogin' %}"><span class="fas
fa-sign-in-alt" aria-hidden="true"></span> Admin Login</a></li>
            </ul>
            {% endif %}
        </div>
        <div class="clearfix"></div>
    </div>
</div>
```

```
<div class="container">
    {% block content %}
    {% endblock %}
  </div>
  <!-- Optional JavaScript -->
  <!-- jQuery first, then Popper.js, then Bootstrap JS -->
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/
zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></
script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/
umd/popper.min.js" integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
crossorigin="anonymous"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/
js/bootstrap.min.js" integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
crossorigin="anonymous"></script>
</body>
</html>
```

-> ADMINBASE.HTML

```
<!doctype html>
<html lang="en">
<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">

  <!-- Bootstrap CSS -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/
bootstrap/4.5.0/css/bootstrap.min.css"
integrity="sha384-9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxF
fc+NcPb1dKGj7Sk" crossorigin="anonymous">

  <title>Blood Bank</title>
</head>
<div class="header">
  <div class="container">
      <div >
          <a><h1 style = "text-align: center; color : rgb(15, 14,
14) ; font-weight : bold">Blood Bank Management System</h1></a>
      </div>
      <div class="top-menu">
<body>
  <nav class="navbar navbar-expand-lg navbar-dark bg-danger">
    <a class="navbar-brand" href="{% url 'adminlogged' %}">Blood
Bank Admin</a>
    <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarNavAltMarkup" aria-
controls="navbarNavAltMarkup" aria-expanded="false" aria-
label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
```

```
 <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
      <div class="navbar-nav">

        <a class="nav-item nav-link active" href="{% url 
'adminlogged' %}">Home</a>
        <a class="nav-item nav-link active" href="{% url 
'allrequest' %}">See All Requests</a>
       <a class="nav-item nav-link active" href="{% url 'home' 
%}">Admin Logout</a>

      </div>
    </div>
  </nav>
</div>
        <div class="clearfix"></div>
    </div>
 <br>
  <div class="container">
    {% block content %}
    {% endblock %}
  </div>
  <!-- Optional JavaScript -->
  <!-- jQuery first, then Popper.js, then Bootstrap JS -->
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" 
integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/
zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></
script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/
umd/popper.min.js" integrity="sha384-
Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo" 
crossorigin="anonymous"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/
js/bootstrap.min.js" integrity="sha384-
OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI" 
crossorigin="anonymous"></script>
</body>
</html>
```

-> FOOTER.HTML

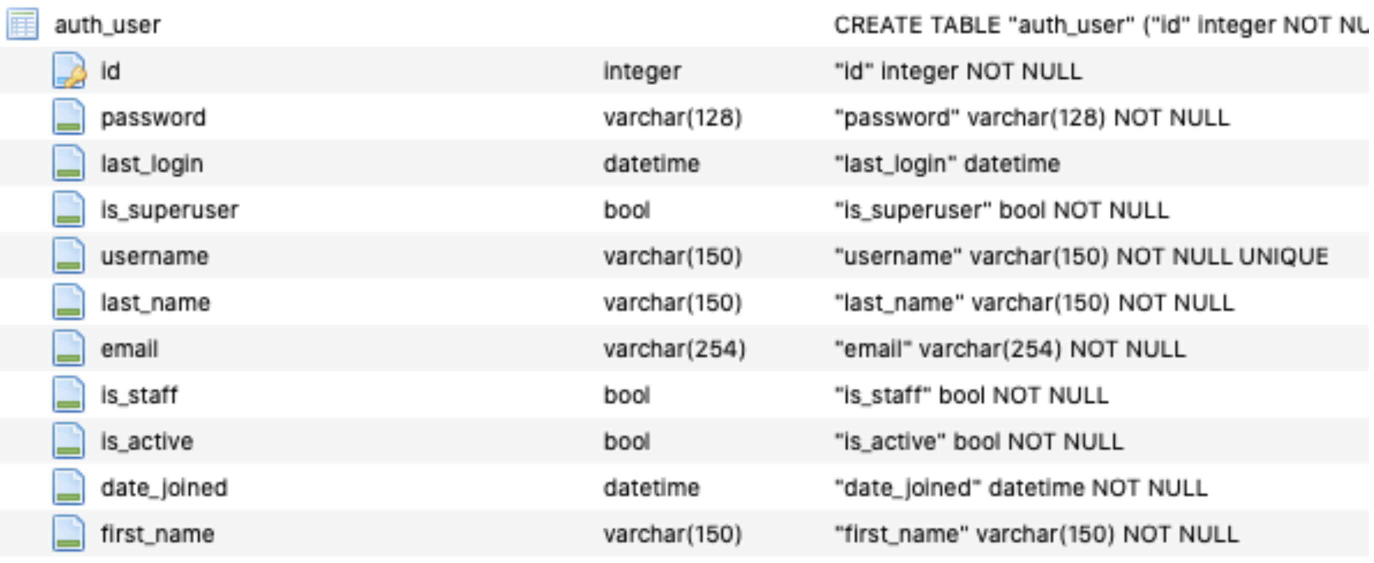
```
{% load static %}

<div class="footer">
     <div class="container">

         <div class="copywrite">
             <p>Blood Bank Management System</p>

         <h5>SIR M VISVESVARAYA INSTITUTE OF TECHNOLOGY</h5>
         <h5>BENGALURU </h5>
         <h5>PHONE :- 9876543210 </h5>
         <h5>EMAIL :- bloodbank@gmail.com </h5>
        </div>



         <div class="social-icons">
              <a href="https://twitter.com"><i class="twitter"></i></
a>
              <a href="https://www.facebook.com"><i
class="facebook"></i></a>

         </div>
         <div class="clearfix"></div>
     </div>
     </div>
{% block body %}

{% endblock %}
```

-> ADMINHOME.HTML

```
{% extends "blood_app/adminbase.html" %}

{% block content %}
  <h1>Available Blood Group</h1>


  <main role="main">

    <div class="album py-5 bg-light">
      <div class="container">
        <div class="row">
          {% for i in all_group %}
          <div class="col-md-4">
            <div class="card mb-4 shadow-sm">
              <div class="card-body">
                <h3 class="card-text">Blood Group: {{i.name}}</h3>
                <h3 class="card-text">Total Donor: {{i.total}}</h3>

                <div class="d-flex justify-content-between align-
items-center">
                  <div class="btn-group">
                    <a href="{% url 'group' i.id %}" class="btn
btn-sm btn-outline-success">See All</a>
                  </div>
                </div>
              </div>
            </div>
          </div>
          {% endfor %}
        </div>
      </div>
    </div>
  </main>
{% endblock %}
```

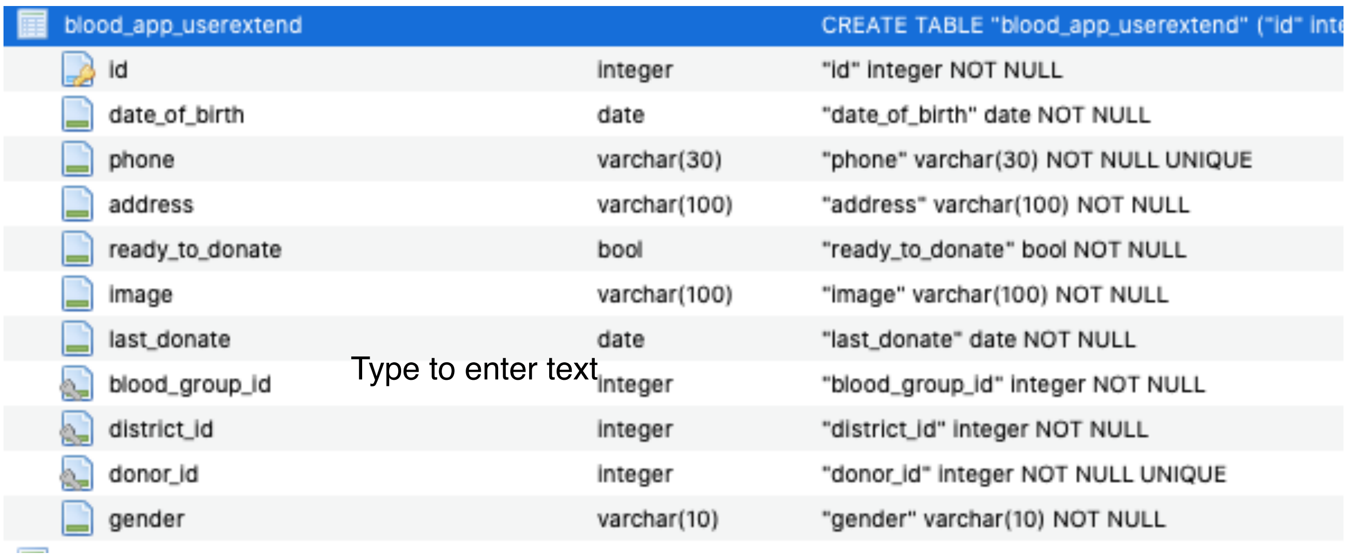
DATABASE TABLES

-USERS TABLE



auth_user		CREATE TABLE "auth_user" ("id" integer NOT NU
id	integer	"id" integer NOT NULL
password	varchar(128)	"password" varchar(128) NOT NULL
last_login	datetime	"last_login" datetime
is_superuser	bool	"is_superuser" bool NOT NULL
username	varchar(150)	"username" varchar(150) NOT NULL UNIQUE
last_name	varchar(150)	"last_name" varchar(150) NOT NULL
email	varchar(254)	"email" varchar(254) NOT NULL
is_staff	bool	"is_staff" bool NOT NULL
is_active	bool	"is_active" bool NOT NULL
date_joined	datetime	"date_joined" datetime NOT NULL
first_name	varchar(150)	"first_name" varchar(150) NOT NULL

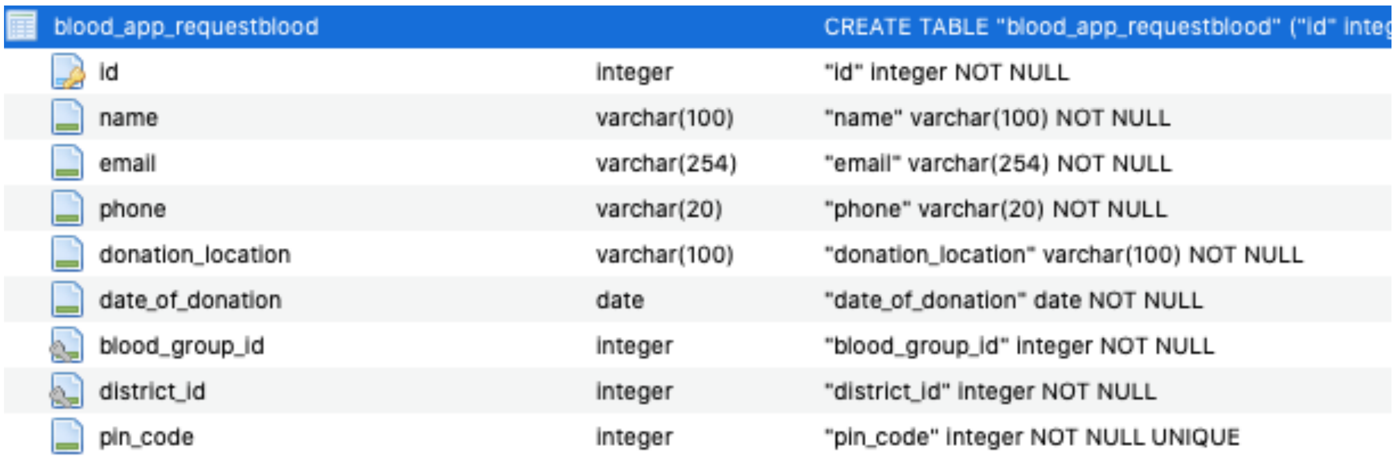
-USERSEXTEND



blood_app_userextend		CREATE TABLE "blood_app_userextend" ("id" inte
id	integer	"id" integer NOT NULL
date_of_birth	date	"date_of_birth" date NOT NULL
phone	varchar(30)	"phone" varchar(30) NOT NULL UNIQUE
address	varchar(100)	"address" varchar(100) NOT NULL
ready_to_donate	bool	"ready_to_donate" bool NOT NULL
image	varchar(100)	"image" varchar(100) NOT NULL
last_donate	date	"last_donate" date NOT NULL
blood_group_id	integer	"blood_group_id" integer NOT NULL
district_id	integer	"district_id" integer NOT NULL
donor_id	integer	"donor_id" integer NOT NULL UNIQUE
gender	varchar(10)	"gender" varchar(10) NOT NULL

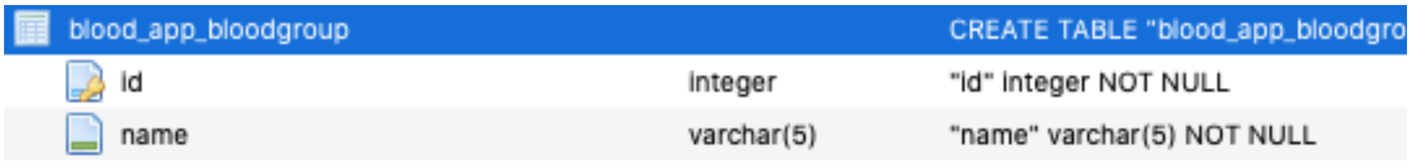
Type to enter text

-REQUESTBLOOD TABLE



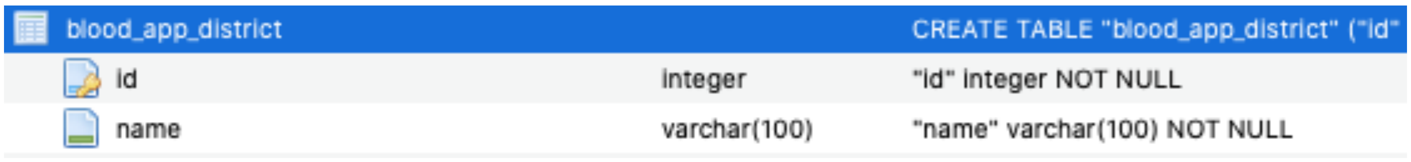
blood_app_requestblood		CREATE TABLE "blood_app_requestblood" ("id" integ
id	integer	"id" integer NOT NULL
name	varchar(100)	"name" varchar(100) NOT NULL
email	varchar(254)	"email" varchar(254) NOT NULL
phone	varchar(20)	"phone" varchar(20) NOT NULL
donation_location	varchar(100)	"donation_location" varchar(100) NOT NULL
date_of_donation	date	"date_of_donation" date NOT NULL
blood_group_id	integer	"blood_group_id" integer NOT NULL
district_id	integer	"district_id" integer NOT NULL
pin_code	integer	"pin_code" integer NOT NULL UNIQUE

-BLOOD GROUP TABLE



blood_app_bloodgroup		CREATE TABLE "blood_app_bloodgro
id	integer	"id" integer NOT NULL
name	varchar(5)	"name" varchar(5) NOT NULL

-DISTRICTS TABLE



blood_app_district		CREATE TABLE "blood_app_district" ("id"
id	integer	"id" integer NOT NULL
name	varchar(100)	"name" varchar(100) NOT NULL

CHAPTER 8

8.SYSTEM TESTING

8.1 Introduction

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

Some prefer saying Software testing as a White Box and Black Box Testing. In simple terms, Software Testing means the Verification of Application Under Test (AUT). This tutorial introduces testing software to the audience and justifies its importance.

8.2 Types of Testing

1. Unit Testing

It focuses on the smallest unit of software design. In this, we test an individual unit or group of interrelated units. It is often done by the programmer by using sample input and observing its corresponding outputs.
Example:
a) In a program we are checking if loop, method or function is working fine.
b) Misunderstood or incorrect, arithmetic precedence.
c) Incorrect initialization

2. Integration Testing

The objective is to take unit tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components is combined to produce output.
Integration testing is of four types: (i) Top-down (ii) Bottom-up (iii) Sandwich (iv) Big-Bang
Example
(a) Black Box testing:- It is used for validation. In this we ignore internal working mechanism and focus on what is the output?. 25
(b) White Box testing:- It is used for verification. In this we focus on internal mechanism i.e. how the output is achieved?

3. Regression Testing

Every time a new module is added leads to changes in the program. This type of testing makes sure that the whole component works properly even after adding components to the complete program.
Example
In school record suppose we have module staff, students and finance combining these modules and checking if on integration these module works fine is regression testing

4. Smoke Testing

This test is done to make sure that software under testing is ready or stable for further testing
It is called a smoke test as the testing an initial pass is done to check if it did not catch the fire or smoke in the initial switch on.
Example:
If project has 2 modules so before going to module make sure that module 1 works properly

5. Alpha Testing

This is a type of validation testing. It is a type of *acceptance testing* which is done before the product is released to customers. It is typically done by QA people.
Example:
When software testing is performed internally within the organization.

6. Beta Testing

The beta test is conducted at one or more customer sites by the end-user of the software. This version is released for a limited number of users for testing in a real-time environment
Example:
When software testing is performed for the limited number of people 26

7. System Testing

This software is tested such that it works fine for the different operating systems. It is covered under the black box testing technique. In this, we just focus on the required input and output without focusing on internal working.
In this, we have security testing, recovery testing, stress testing, and performance testing
Example:
This include functional as well as non functional testing.

8. Stress Testing

In this, we give unfavorable conditions to the system and check how they perform in those conditions.
Example:
(a) Test cases that require maximum memory or other resources are executed
(b) Test cases that may cause thrashing in a virtual operating system
(c) Test cases that may cause excessive disk requirement

9. Performance Testing

It is designed to test the run-time performance of software within the context of an integrated system. It is used to test the speed and effectiveness of the program. It is also called load testing. In it we check, what is the performance of the system in the given load.
Example:
Checking number of processor cycles.

10. Object-Oriented Testing:

This testing is a combination of various testing techniques that help to verify and validate object-oriented software. This testing is done in the following manner:

● Testing of Requirements,

● Design and Analysis of Testing,

● Testing of Code,

● Integration testing,

● System testing,

8.3 Levels of Testing

Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are −

- Functional Testing

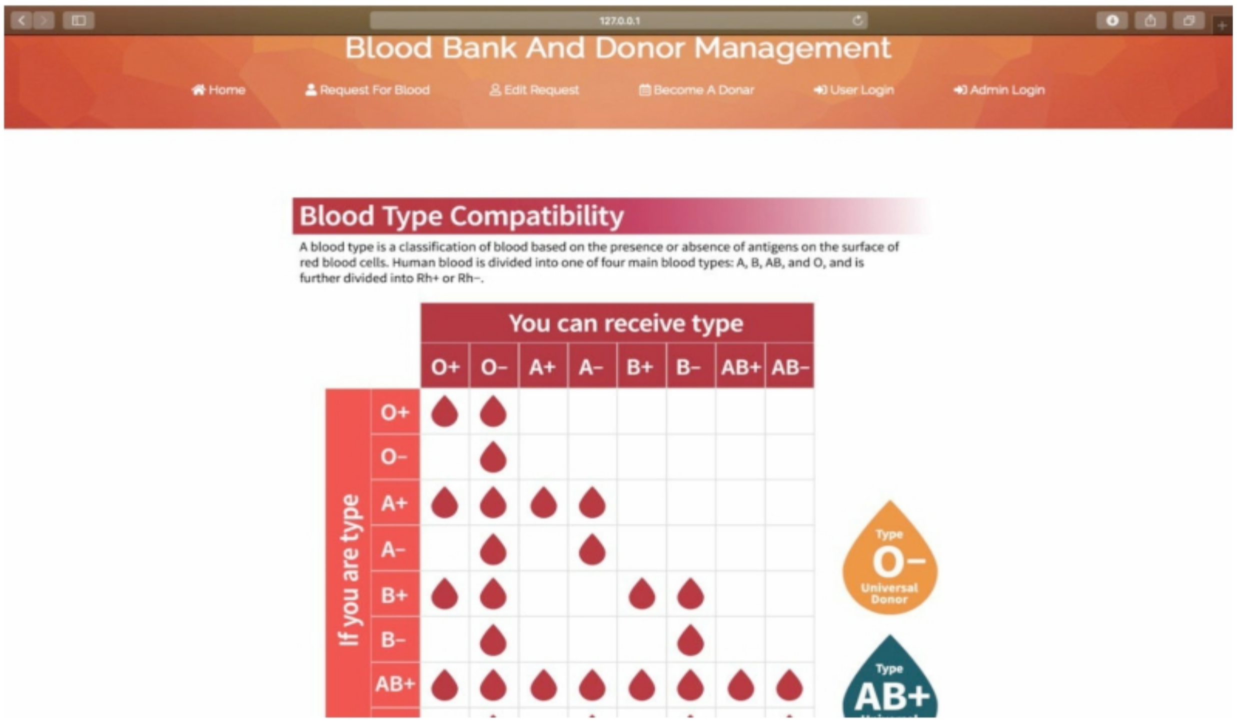
  This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of a software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

- Non-Functional Testing

  It is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.
  An excellent example of non-functional test would be to check how many people can simultaneously login into a software.
  Non-functional testing is equally important as functional testing and affects client satisfaction.

CHAPTER 9
RESULTS AND SCREENSHOTS

HOME PAGE

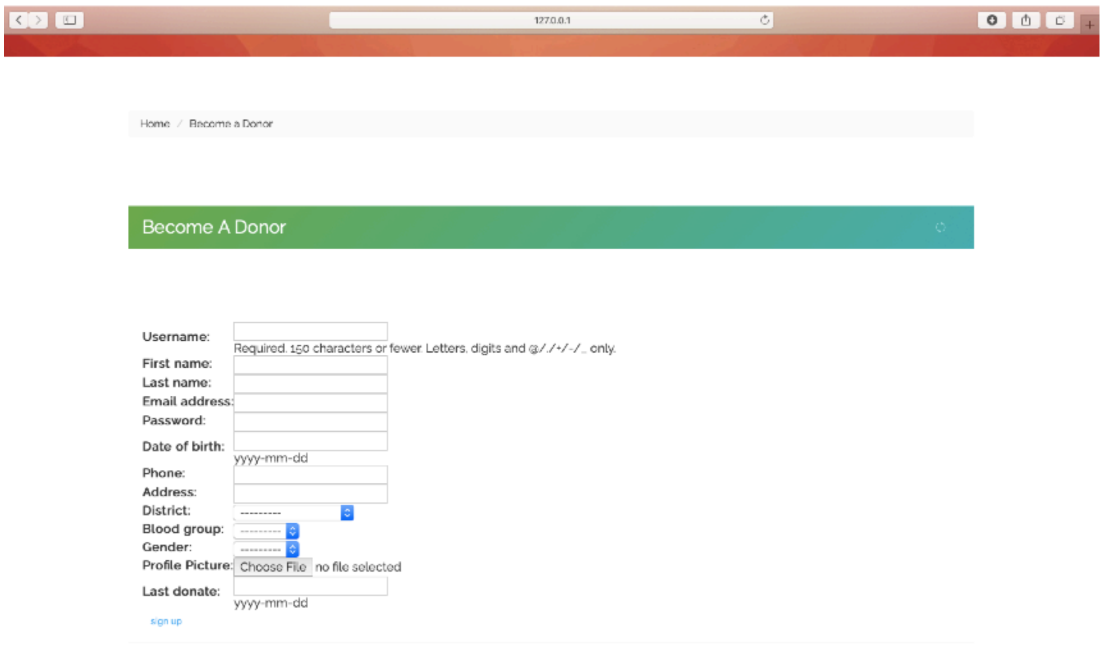


Blood Type Compatibility

A blood type is a classification of blood based on the presence or absence of antigens on the surface of red blood cells. Human blood is divided into one of four main blood types: A, B, AB, and O, and is further divided into Rh+ or Rh−.

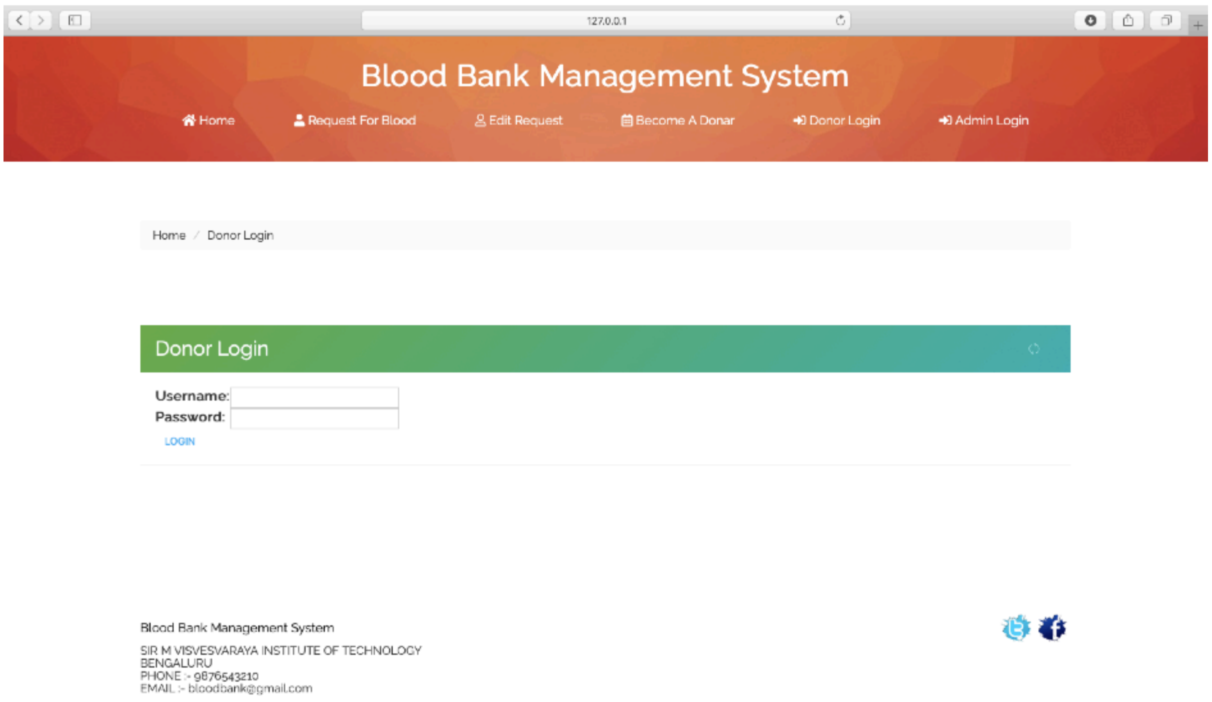
	You can receive type							
If you are type	O+	O−	A+	A−	B+	B−	AB+	AB−
O+	●	●						
O−		●						
A+	●	●	●	●				
A−		●		●				
B+	●	●			●	●		
B−		●				●		
AB+	●	●	●	●	●	●	●	●

SIGN UP PAGE



Username:	Required. 150 characters or fewer Letters, digits and @/./+/-/_ only.
First name:	
Last name:	
Email address:	
Password:	
Date of birth:	yyyy-mm-dd
Phone:	
Address:	
District:	---------
Blood group:	---------
Gender:	---------
Profile Picture:	Choose File no file selected
Last donate:	yyyy-mm-dd

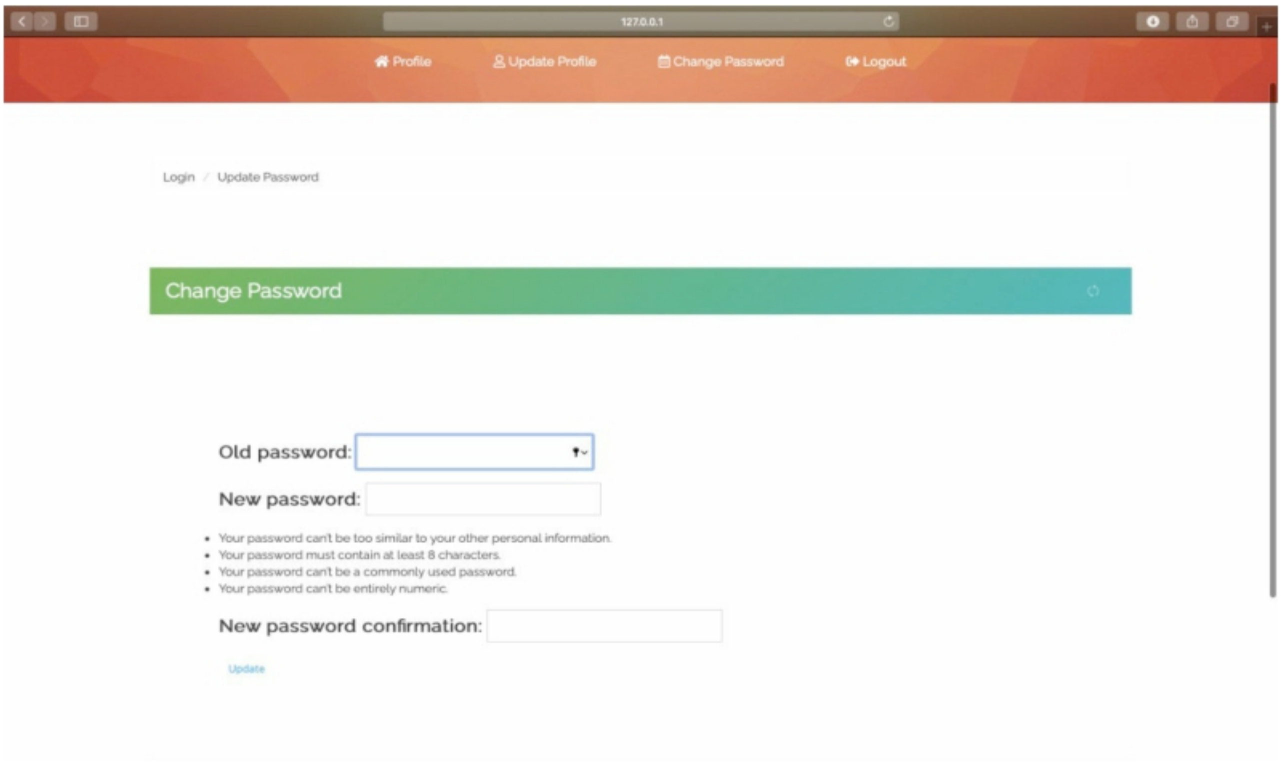
DONOR LOGIN



Username:
Password:

Blood Bank Management System
SIR M VISVESVARAYA INSTITUTE OF TECHNOLOGY
BENGALURU
PHONE :- 9876543210
EMAIL :- bloodbank@gmail.com

CHANGE PASSWORD PAGE

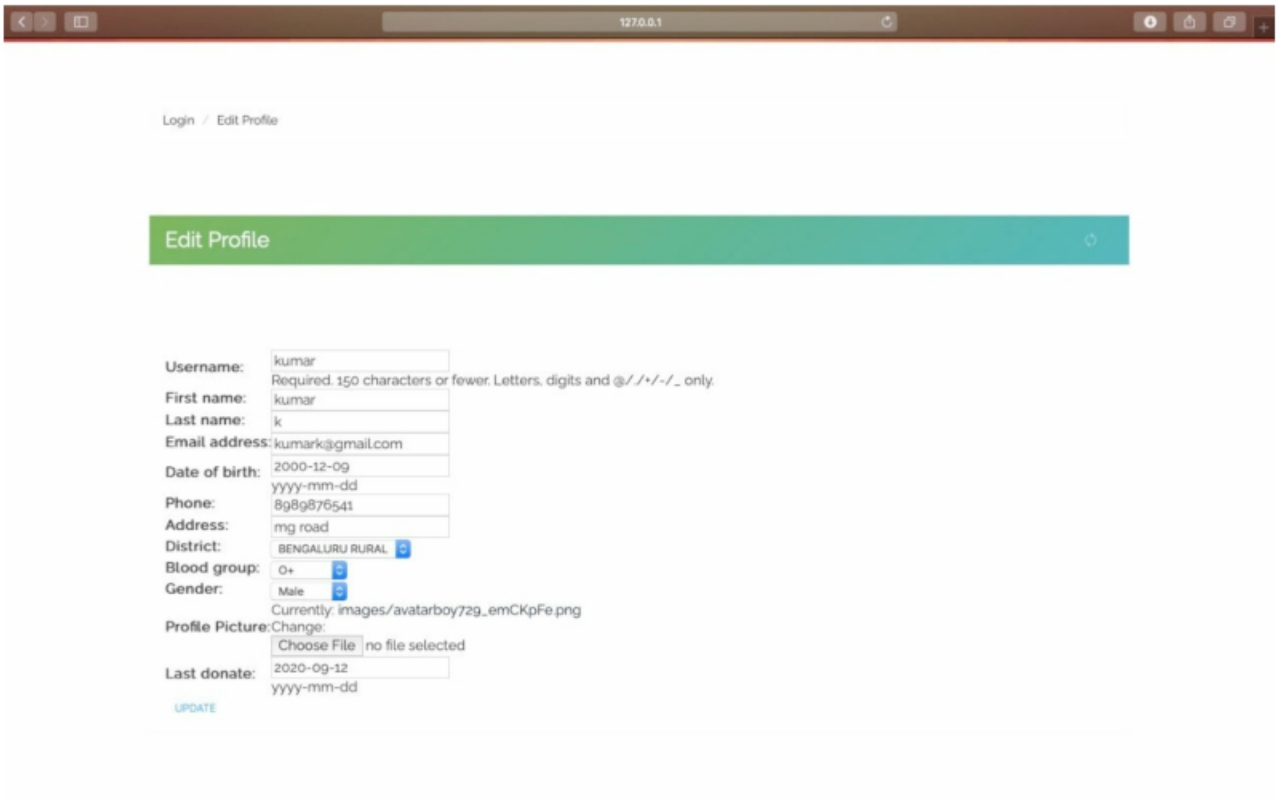


Old password:
New password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

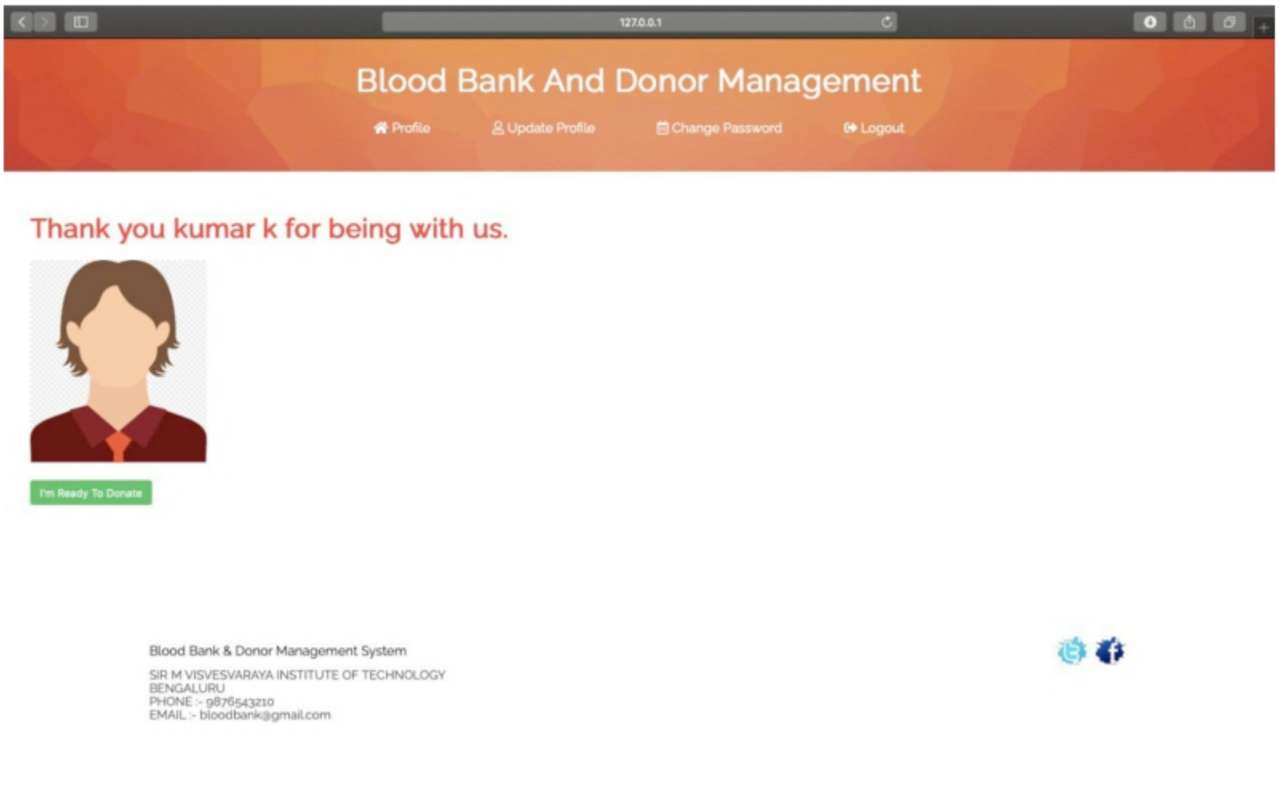
New password confirmation:

EDIT PROFILE



Username: kumar
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.
First name: kumar
Last name: k
Email address: kumark@gmail.com
Date of birth: 2000-12-09
yyyy-mm-dd
Phone: 8989876541
Address: mg road
District: BENGALURU RURAL
Blood group: O+
Gender: Male
Currently: images/avatarboy729_emCKpFe.png
Profile Picture: Change:
Choose File no file selected
Last donate: 2020-09-12
yyyy-mm-dd
UPDATE

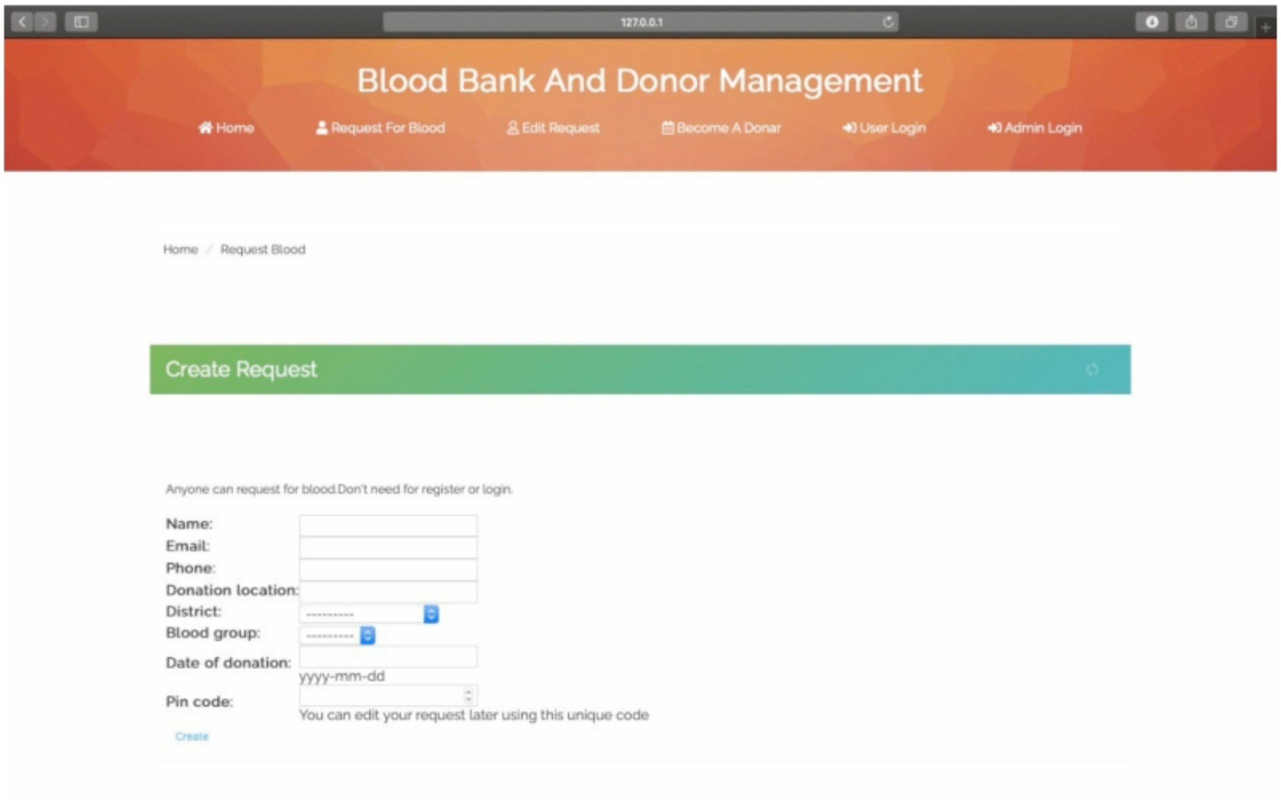
DONOR PROFILE PAGE



Thank you kumar k for being with us.

Blood Bank & Donor Management System
SIR M VISVESVARAYA INSTITUTE OF TECHNOLOGY
BENGALURU
PHONE :- 9876543210
EMAIL :- bloodbank@gmail.com

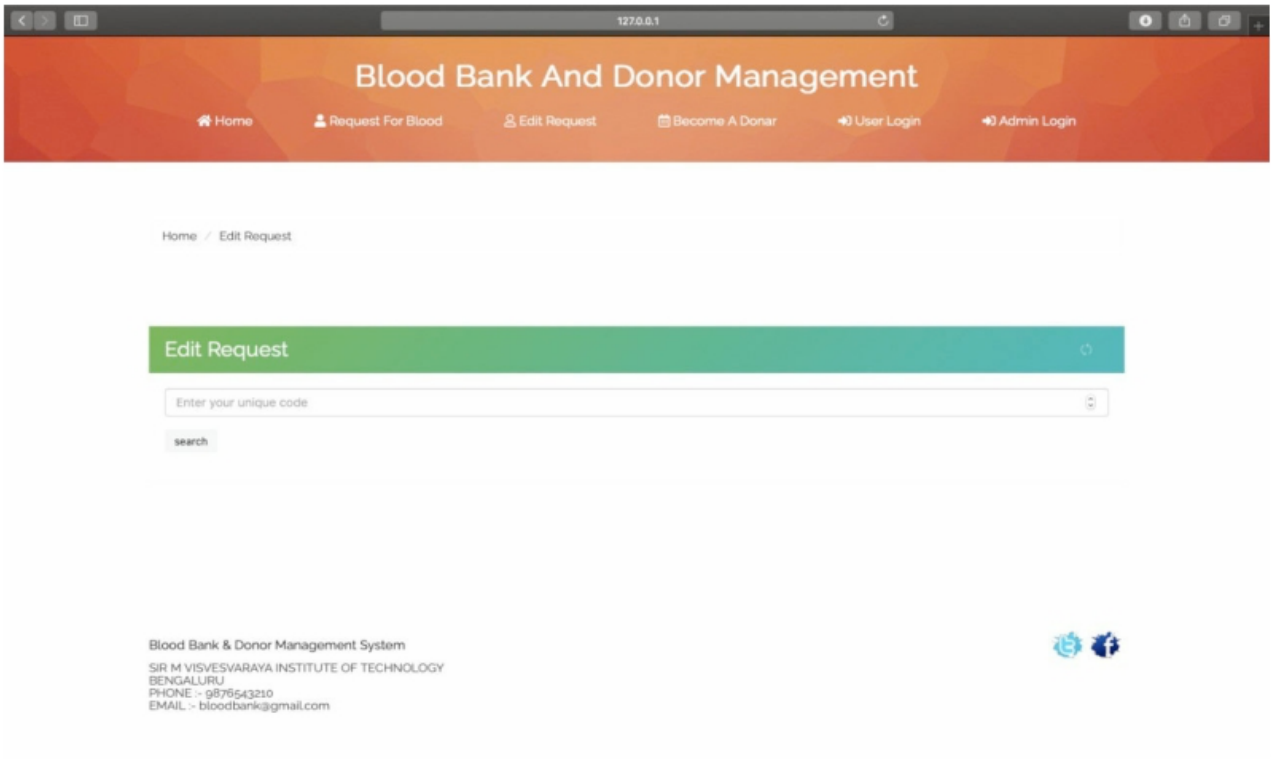
CREATE REQUEST PAGE



Anyone can request for blood.Don't need for register or login.

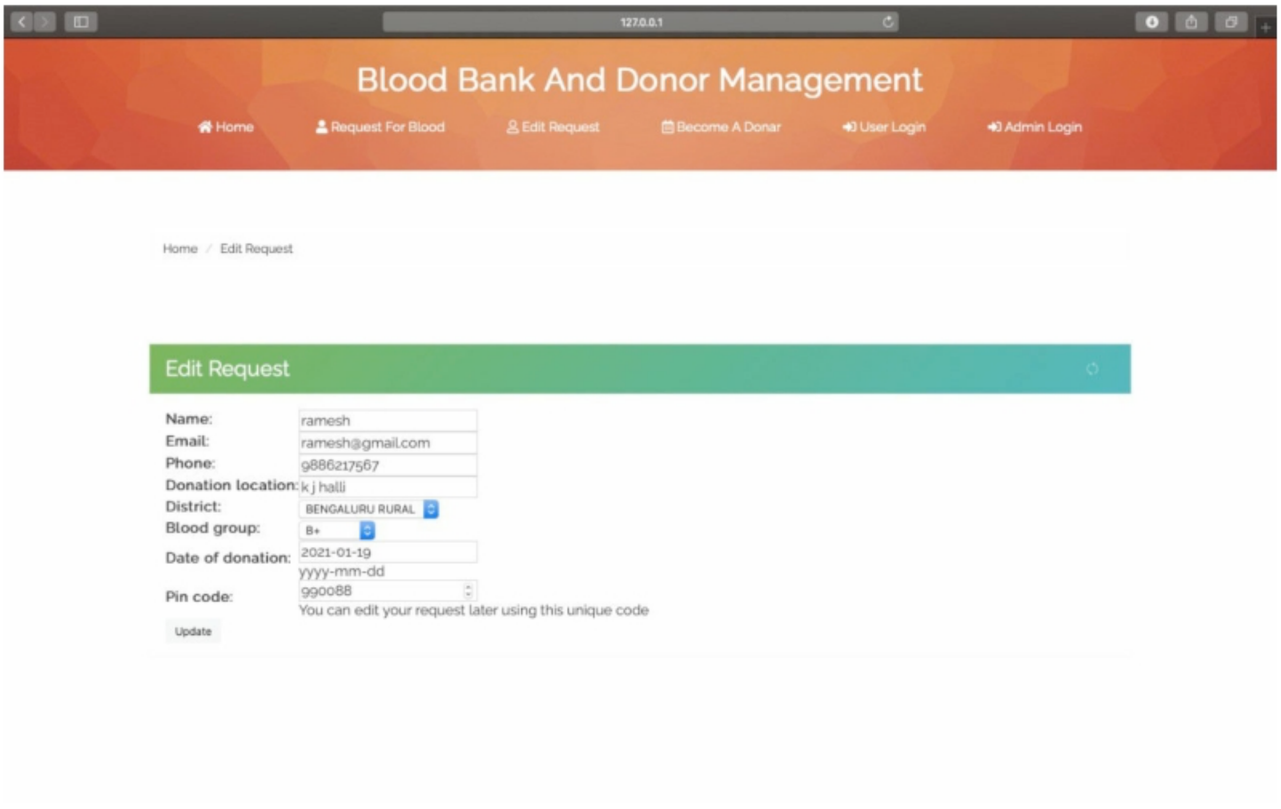
Name:
Email:
Phone:
Donation location:
District: ----------
Blood group: ----------
Date of donation: yyyy-mm-dd
Pin code: You can edit your request later using this unique code

EDIT REQUEST PAGE



Blood Bank & Donor Management System

SIR M VISVESVARAYA INSTITUTE OF TECHNOLOGY
BENGALURU
PHONE :- 9876543210
EMAIL :- bloodbank@gmail.com


127.0.0.1

Blood Bank And Donor Management

Home | Request For Blood | Edit Request | Become A Donar | User Login | Admin Login

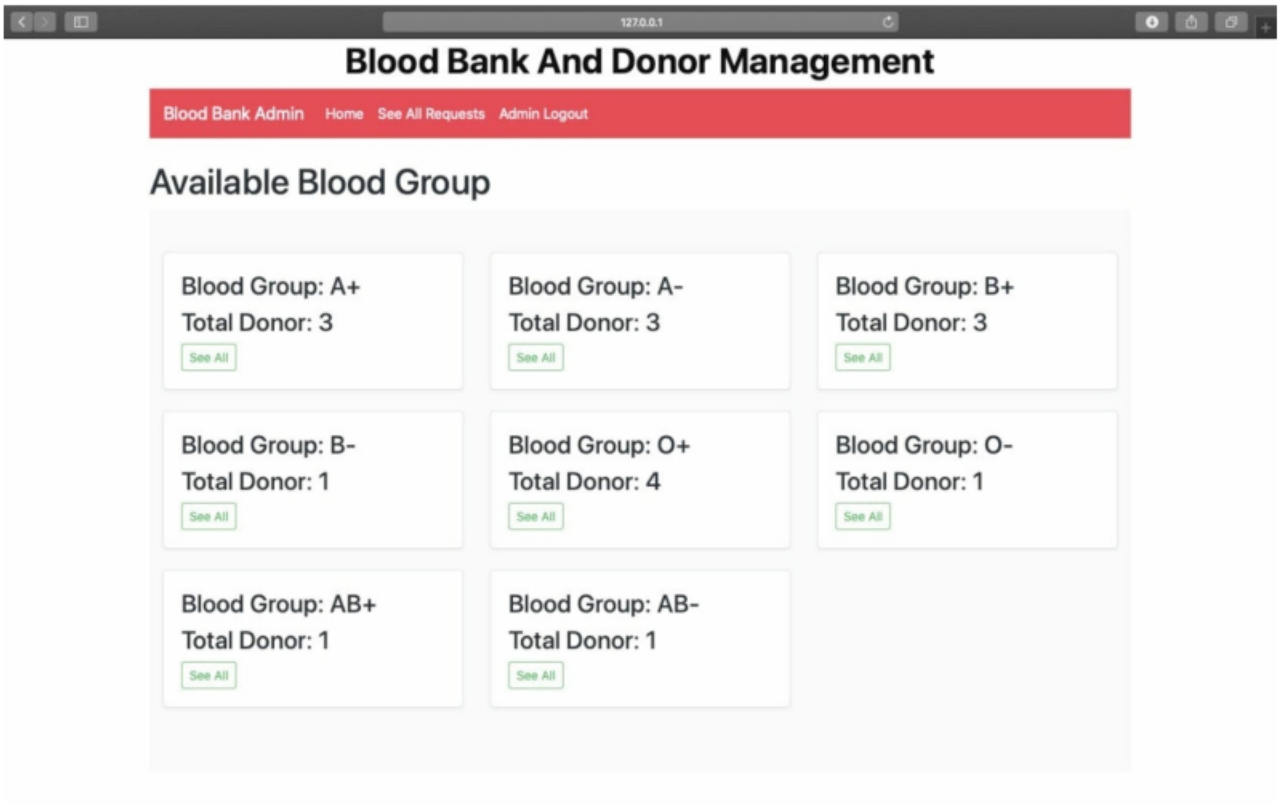
Home / Edit Request

Edit Request

Name:	ramesh
Email:	ramesh@gmail.com
Phone:	9886217567
Donation location:	k j halli
District:	BENGALURU RURAL
Blood group:	B+
Date of donation:	2021-01-19 yyyy-mm-dd
Pin code:	990088 You can edit your request later using this unique code

Update

BLOOD BANK ADMIN HOME


127.0.0.1

Blood Bank And Donor Management

Blood Bank Admin | Home | See All Requests | Admin Logout

Available Blood Group

Blood Group: A+
Total Donor: 3
See All

Blood Group: A-
Total Donor: 3
See All

Blood Group: B+
Total Donor: 3
See All

Blood Group: B-
Total Donor: 1
See All

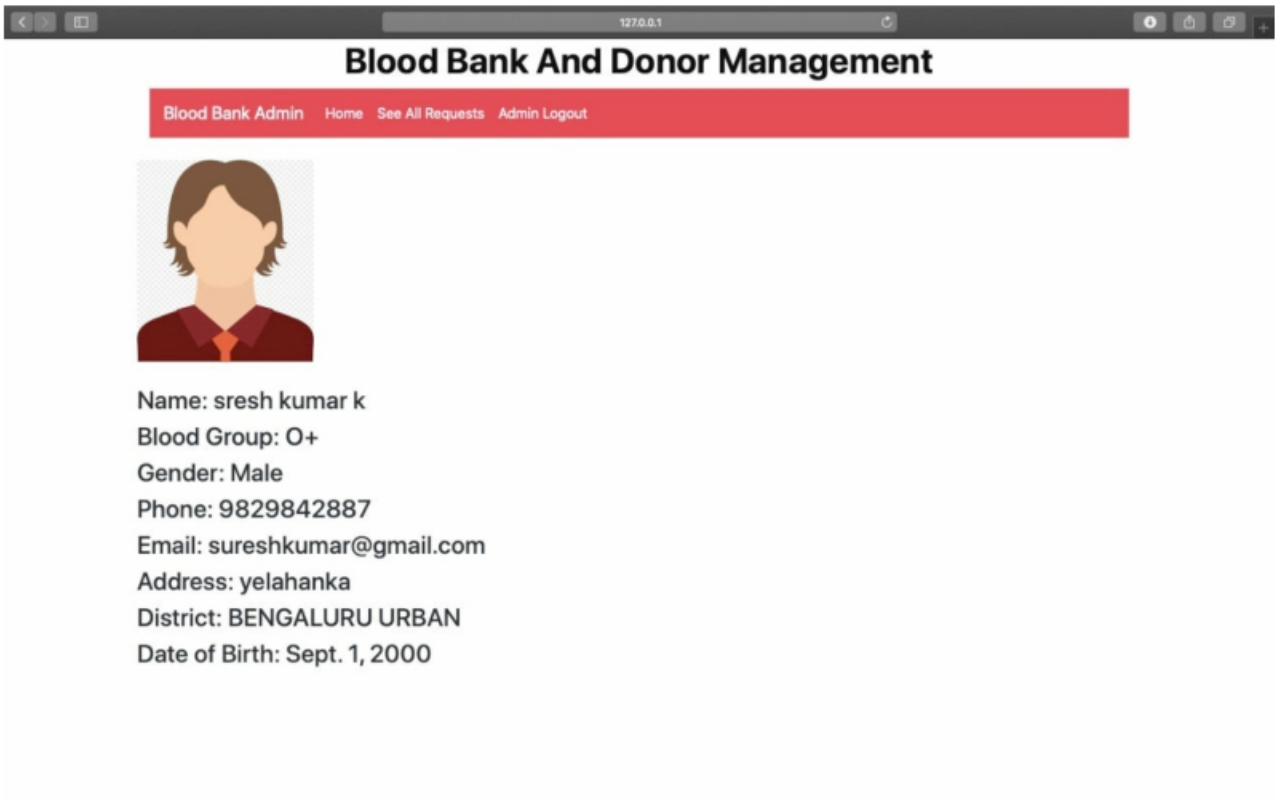
Blood Group: O+
Total Donor: 4
See All

Blood Group: O-
Total Donor: 1
See All

Blood Group: AB+
Total Donor: 1
See All

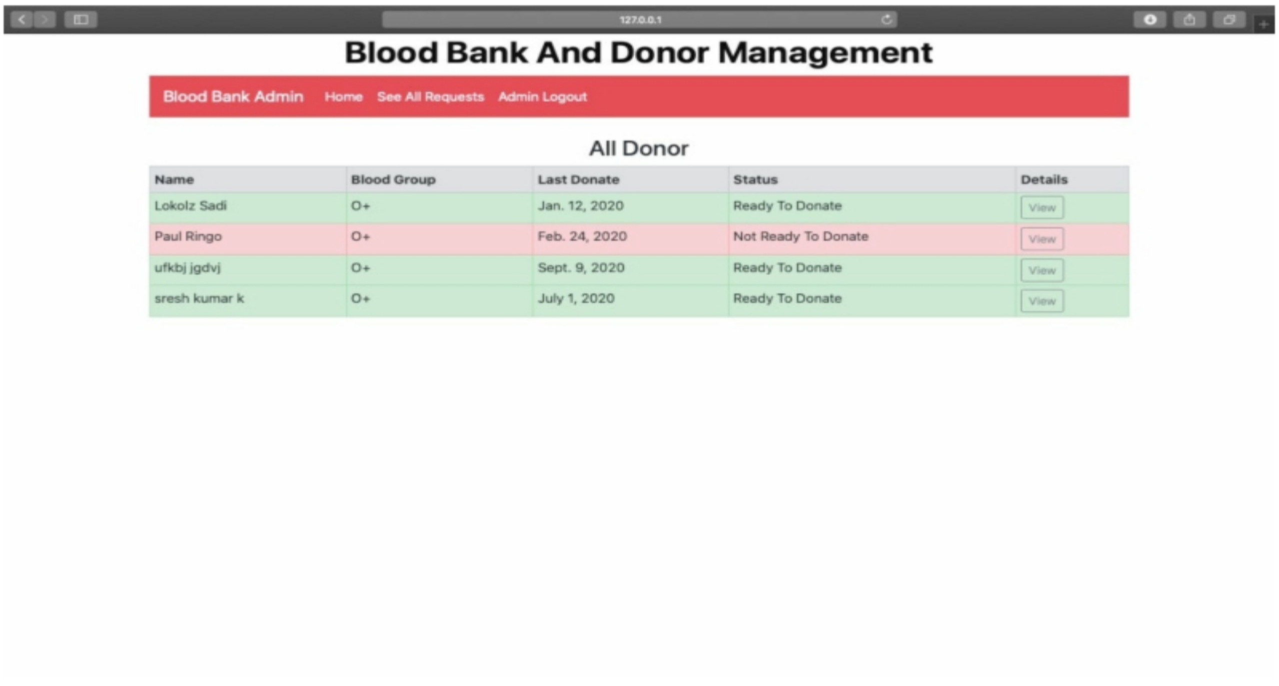
Blood Group: AB-
Total Donor: 1
See All

BLOOD BANK AMIN USER PROFILE



Name: sresh kumar k
Blood Group: O+
Gender: Male
Phone: 9829842887
Email: sureshkumar@gmail.com
Address: yelahanka
District: BENGALURU URBAN
Date of Birth: Sept. 1, 2000

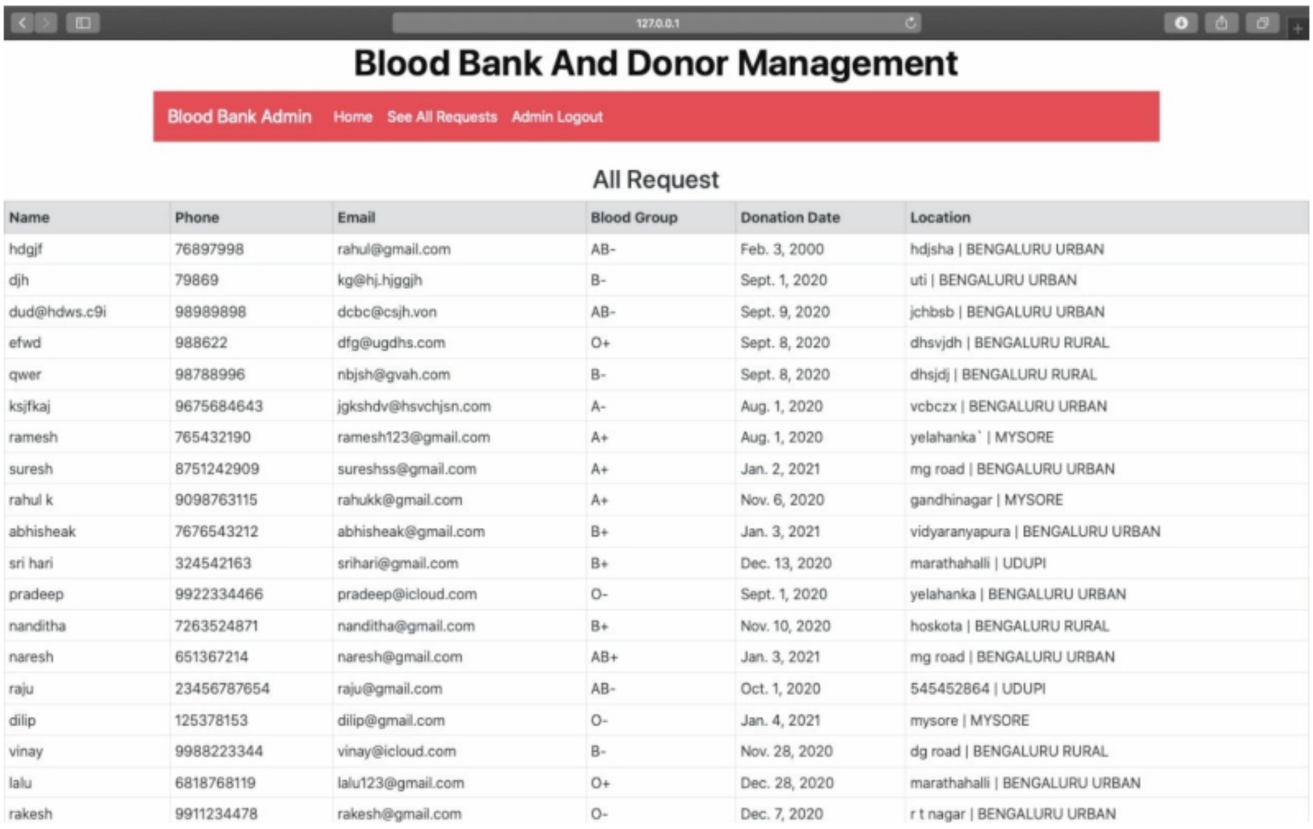
BLOOD BANK ALL DONORS



All Donor

Name	Blood Group	Last Donate	Status	Details
Lokolz Sadi	O+	Jan. 12, 2020	Ready To Donate	View
Paul Ringo	O+	Feb. 24, 2020	Not Ready To Donate	View
ufkbj jgdvj	O+	Sept. 9, 2020	Ready To Donate	View
sresh kumar k	O+	July 1, 2020	Ready To Donate	View

BLOOD BANK ALL REQUESTS



All Request

Name	Phone	Email	Blood Group	Donation Date	Location
hdgjf	76897998	rahul@gmail.com	AB-	Feb. 3, 2000	hdjsha | BENGALURU URBAN
djh	79869	kg@hj.hjggjh	B-	Sept. 1, 2020	uti | BENGALURU URBAN
dud@hdws.c9i	98989898	dcbc@csjh.von	AB-	Sept. 9, 2020	jchbsb | BENGALURU URBAN
efwd	988622	dfg@ugdhs.com	O+	Sept. 8, 2020	dhsvjdh | BENGALURU RURAL
qwer	98788996	nbjsh@gvah.com	B-	Sept. 8, 2020	dhsjdj | BENGALURU RURAL
ksjfkaj	9675684643	jgkshdv@hsvchjsn.com	A-	Aug. 1, 2020	vcbczx | BENGALURU URBAN
ramesh	765432190	ramesh123@gmail.com	A+	Aug. 1, 2020	yelahanka` | MYSORE
suresh	8751242909	sureshss@gmail.com	A+	Jan. 2, 2021	mg road | BENGALURU URBAN
rahul k	9098763115	rahukk@gmail.com	A+	Nov. 6, 2020	gandhinagar | MYSORE
abhisheak	7676543212	abhisheak@gmail.com	B+	Jan. 3, 2021	vidyaranyapura | BENGALURU URBAN
sri hari	324542163	srihari@gmail.com	B+	Dec. 13, 2020	marathahalli | UDUPI
pradeep	9922334466	pradeep@icloud.com	O-	Sept. 1, 2020	yelahanka | BENGALURU URBAN
nanditha	7263524871	nanditha@gmail.com	B+	Nov. 10, 2020	hoskota | BENGALURU RURAL
naresh	651367214	naresh@gmail.com	AB+	Jan. 3, 2021	mg road | BENGALURU URBAN
raju	23456787654	raju@gmail.com	AB-	Oct. 1, 2020	545452864 | UDUPI
dilip	125378153	dilip@gmail.com	O-	Jan. 4, 2021	mysore | MYSORE
vinay	9988223344	vinay@icloud.com	B-	Nov. 28, 2020	dg road | BENGALURU RURAL
lalu	6818768119	lalu123@gmail.com	O+	Dec. 28, 2020	marathahalli | BENGALURU URBAN
rakesh	9911234478	rakesh@gmail.com	O-	Dec. 7, 2020	r t nagar | BENGALURU URBAN

CHAPTER 10

10. CONCLUSION AND SCOPE

Blood and its components are a very important resource and hence should be used in a justifiable manner. The pattern of utilization and wastage needs to be recorded and a real time database is needed to monitor the need and consumption of blood to minimize its wastage. This can helps in minimizing wastage by reallocation of reserved but un utilized blood components. A proper inventory management system with accurate and timely database formation is necessary to minimize wastage. By making sure that blood is utilized to its maximum we can make sure that many more lives are saved .

CHAPTER 11

11. References and web links

11.1 References:

· Manmohan Singhal, Manish Patel , Devesh Kapoor and Dalal Mitta (2013) “A research analysis on blood component usage and wastage in blood bank and blood component center”

· N. Adarsh, J. Arpitha, M. D. Ali, N. M. Charan and P. G. Mahendrakar, "Effective blood bank management based on RFID in real time systems," *2014 International Conference on Embedded Systems (ICES)*, Coimbatore, 2014, pp. 287-290, doi: 10.1109/ EmbeddedSys.2014.6953176.

· Global status report on blood safety and availability 2016. Geneva: World Health Organization; 2017. Licence: CC BY-NC-SA 3.0 IGO.

· Ibrahim Fawze Akar, Tukur Anas Mohammad, Mohamed Ismail Z, “CBBR Centralized Blood Bank Repository”

· Sulaiman, S. , Abdul Hamid, A. , & Najihah Yusri, N. (2015). Development of a Blood Bank Management System. Procedia - Social and Behavioral Sciences, 195 . doi: 10.1016/j.sbspro.2015.06.215

· Hazzazi, N. , Wijesekera, D. , & Hindawi, S. (2014). Formalizing and Verifying Workflows Used in Blood Banks. Procedia Technology, 16 . doi: 10.1016/ j.protcy.2014.10.143

· Selvamani, K. , & Kumar Rai, A. (2015). A Novel Technique for Online Blood Bank Management. Procedia Computer Science, 48 . doi: 10.1016/j.procs. 2015.04.137

· Duong, L. , Wood, L. , & Wang, W. (2015). A Multi-criteria Inventory Management System for Perishable & Substitutable Products. Procedia Manufacturing, 2 . doi: 10.1016/j.promfg.2015.07.012

· Atan, Z. , & Rousseau, M. (2015). Inventory optimization for perishables subject to supply disruptions. Optimization Letters, 10 (1). doi: 10.1007/ s11590-015-0858-7

· Wong, K. (2011). Virtual blood bank. Journal of Pathology Informatics, 2 (1). doi: 10.4103/2153-3539.76155

· Todorovic, V. , Neag, M. , & Lazarevic, M. (2014). On the Usage of RFID Tags for Tracking and Monitoring of Shipped Perishable Goods. Procedia Engineering, 69 . doi: 10.1016/j.proeng.2014.03.127

11.2 Weblinks:

https://www.w3schools.com/html/ https://
www.w3schools.com/css/ https://
www.w3schools.com/bootstrap4 https://
www.tutorialspoint.com/django/index.html https://
www.w3schools.com/js/DEFAULT.asp