BLOOD DONATION & REQUEST PORTAL

A MINI-PROJECT REPORT

Submitted by

DHEJASVI J B **241801053**

DIVYA LAKSHMI D **241801060**

in partial fulfillment of the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE & DATA SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project **"BLOOD DONATION & REQUEST PORTAL"** is the bonafide work of **"DHEJASVI J B & DIVYA LAKSHMI D"** who carried out the project work under my supervision.

SIGNATURE

R. SAVITHRI

ASSISTANT PROFESSOR SG

Dept of Artificial Intelligence & Data
Science,
Rajalakshmi Engineering College
Chennai

This mini project report is submitted for the viva voce examination to be held on

____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT

The “Blood Donation and Request Portal” is a web-based application developed to bridge the gap between blood donors, patients, and hospitals through a unified platform. It allows patients to raise blood requests instantly, donors to update their availability, and administrators to manage user information and blood bank records efficiently. The system ensures secure user authentication with role-based access for donors, patients, and admins, maintaining accurate records of requests, donations, and notifications.

Developed using Flask (Python) for backend processing and MySQL for reliable data storage, the portal offers real-time notifications and automated blood group matching to ensure quick responses during emergencies. By streamlining the blood donation process and promoting transparency, the project aims to make blood management faster, safer, and more accessible—ultimately contributing to saving lives.

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M.THANGAM MEGANATHAN** for their timely support and encouragement.

We are greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended to us by our Head Of The Department **Dr. E.M. MALATHY** and our Deputy Head Of The Department **Dr. J. MANORANJINI** for being ever supporting force during our project work

We also extend our sincere and hearty thanks to our internal guide **Dr. V. JANANEE ,** for her valuable guidance and motivation during the completion of this project.

Our sincere thanks to our family members, friends and other staff members of computer science engineering.

1. DHEJASVI J B

2. DIVYA LAKSHMI D

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1 – INTRODUCTION

1.1 INTRODUCTION

The Blood Donation and Request Portal is a web-based platform designed to connect blood donors, patients, and hospitals through a centralized system. It simplifies the process of finding donors during emergencies and ensures quick, accurate matching of blood groups. The portal provides secure login for all users, manages donor availability, and sends real-time notifications for urgent requests, ensuring timely and reliable blood supply.

1.2 SCOPE OF THE WORK

The project enables secure registration, login, and role-based access for donors, patients, and admins. Patients can create blood requests, donors can update their availability, and admins can manage user data and blood banks. The system is built using Python (Django) for backend processing, MySQL for data storage, and HTML/CSS for a responsive user interface.

1.3 PROBLEM STATEMENT

During medical emergencies, finding suitable blood donors is often slow due to the absence of a unified platform. Manual tracking causes delays, communication gaps, and record mismatches, risking patient lives. A digital system is required to automate donor matching, manage data efficiently, and provide instant updates to all users.

1.4 AIM AND OBJECTIVES OF THE PROJECT

Aim:

To develop a secure and efficient portal that streamlines blood donation and request management.

Objectives:

- To connect donors and patients through a single online platform.
- To enable role-based access for donors, patients, and admins.
- To automate blood request matching by group and location.
- To maintain clear records of donations and requests.
- To send real-time alerts for urgent blood needs.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Processor	:	Intel i5
Memory Size	:	8GB (Minimum)
HDD	:	1 TB (Minimum)

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS 10
Front – End	:	Python
Back - End	:	MySql
Language	:	python,SQL

CHAPTER 3

MODULE DESCRIPTION

This application consists of three main modules. When the program runs, the home page provides options for Admin, Login, and Register. Each module has its own specific functionalities as described below.

1. Admin Module

The Admin has complete control over the system. The administrator can log in using a valid username and password. Once logged in, the admin can manage user accounts, oversee donor and patient information, handle blood bank data, and monitor blood requests and donations. The admin also ensures that all operations are properly maintained and updated in the database.

2. Login Module

The Login module allows existing users — whether donors or patients — to access their accounts securely. Users must enter their registered email and password to log in.

Donors can update their availability status, view donation history, and respond to blood requests.

Patients can create new blood requests, track their status, and view notification.

CHAPTER 4

SAMPLE CODING

```
# app.py - Complete Blood Donation Portal


from flask import Flask, render_template, request, redirect, url_for, session, flash

from werkzeug.security import generate_password_hash, check_password_hash

from config import Config

from db_config import mysql, init_db

from functools import wraps


# Initialize Flask app

app = Flask(__name__)

app.config.from_object(Config)


# Initialize database

init_db(app)


# ------------------------

# Decorators

# ------------------------
```

```
def login_required(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        if 'loggedin' not in session:
            flash('Please login first!', 'warning')
            return redirect(url_for('login'))
        return f(*args, **kwargs)
    return decorated_function


def role_required(role):
    def decorator(f):
        @wraps(f)
        def decorated_function(*args, **kwargs):
            if 'role' not in session or session['role'] != role:
                flash('Access denied!', 'danger')
                return redirect(url_for('index'))
            return f(*args, **kwargs)
        return decorated_function
    return decorator


# ------------------------
# Routes
# -------------------------

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/test-db')
def test_db():
    try:
        cursor = mysql.connection.cursor()
        cursor.execute("SELECT DATABASE();")
        db_name = cursor.fetchone()
        cursor.close()
        return f"Connected to database: {db_name}"
    except Exception as e:
        return f"Database connection failed: {str(e)}"


@app.route('/debug-donors')

def debug_donors():
    try:
        cursor = mysql.connection.cursor()

        cursor.execute('SELECT * FROM Donors')

        donors = cursor.fetchall()

        cursor.close()

        return f"<pre>Donors in database: {donors}</pre>"

    except Exception as e:

        return f"<pre>Error: {str(e)}</pre>"



@app.route('/register', methods=['GET', 'POST'])

def register():

    if request.method == 'POST':

        username = request.form['username']

        email = request.form['email']

        password = request.form['password']

        role = request.form['role']


        hashed_password = generate_password_hash(password)


        try:

            cursor = mysql.connection.cursor()

            cursor.execute('SELECT * FROM Users WHERE email = %s', (email,))
            account = cursor.fetchone()

            if account:
                flash('Account already exists!', 'danger')
                return redirect(url_for('register'))

            cursor.execute(
                'INSERT INTO Users (username, email, password, role) VALUES (%s, %s,
%s, %s)',
                (username, email, hashed_password, role)
            )
            mysql.connection.commit()
            cursor.close()

            flash('Registration successful! Please login.', 'success')
            return redirect(url_for('login'))

        except Exception as e:
            flash(f'Error: {str(e)}', 'danger')
            return redirect(url_for('register'))

    return render_template('register.html')
```

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM Users WHERE email = %s', (email,))
        account = cursor.fetchone()
        cursor.close()

        if account and check_password_hash(account['password'], password):
            session['loggedin'] = True
            session['user_id'] = account['user_id']
            session['username'] = account['username']
            session['role'] = account['role']

            flash('Login successful!', 'success')

            if account['role'] == 'donor':
                return redirect(url_for('donor_dashboard'))
            elif account['role'] == 'patient':
                return redirect(url_for('patient_dashboard'))
            elif account['role'] == 'admin':
                return redirect(url_for('admin_dashboard'))
            else:
                flash('Unknown role!', 'warning')
                return redirect(url_for('login'))
        else:
            flash('Invalid email or password!', 'danger')
            return render_template('login.html')
    else:
        return render_template('login.html')


@app.route('/logout')
def logout():
    session.clear()
    flash('You have been logged out.', 'info')
    return redirect(url_for('login'))


# ------------------------
# ADMIN ROUTES
# ------------------------
```

```
@app.route('/admin/dashboard')

@login_required

@role_required('admin')

def admin_dashboard():

    cursor = mysql.connection.cursor()


    cursor.execute('SELECT COUNT(*) AS total FROM Donors')

    total_donors = cursor.fetchone()['total']


    cursor.execute('SELECT COUNT(*) AS total FROM Patients')

    total_patients = cursor.fetchone()['total']


    cursor.execute('SELECT COUNT(*) AS total FROM blood_request_needed')

    total_requests = cursor.fetchone()['total']


    cursor.execute('SELECT COUNT(*) AS total FROM Donations')

    total_donations = cursor.fetchone()['total']


    stats = {

        'total_donors': total_donors,

        'total_patients': total_patients,
        'total_requests': total_requests,
        'total_donations': total_donations
    }

    cursor.close()
    return render_template('admin_dashboard.html', stats=stats)

@app.route('/admin/all-donors-details')
@login_required
@role_required('admin')

def all_donors_details():
    cursor = mysql.connection.cursor()

    cursor.execute('''
        SELECT d.*, u.username, u.email
        FROM Donors d
        JOIN Users u ON d.user_id = u.user_id
        ORDER BY d.city, d.blood_group
    ''')

    donors = cursor.fetchall()
    cursor.close()

    return render_template('all_donors_details.html', donors=donors)

@app.route('/admin/donors-list')
@login_required
@role_required('admin')
def donors_list():
    cursor = mysql.connection.cursor()

    cursor.execute('''
        SELECT d.*, u.username, u.email
        FROM Donors d
        JOIN Users u ON d.user_id = u.user_id
        ORDER BY d.city, d.blood_group
    ''')

    donors = cursor.fetchall()
    cursor.close()

    return render_template('admin_donors_list.html', donors=donors)
```

```
@app.route('/admin/requests-list')
@login_required
@role_required('admin')
def requests_list():
    cursor = mysql.connection.cursor()

    cursor.execute('''
        SELECT br.*, p.hospital_name, p.city, p.contact_number, u.username, u.email
        FROM blood_request_needed br
        JOIN Patients p ON br.patient_id = p.patient_id
        JOIN Users u ON p.user_id = u.user_id
        ORDER BY br.request_date DESC
    ''')

    requests = cursor.fetchall()
    cursor.close()

    return render_template('admin_requests_list.html', requests=requests)

@app.route('/admin/patients-list')
```

```
@login_required
@role_required('admin')
def patients_list():
    cursor = mysql.connection.cursor()

    cursor.execute('''
        SELECT p.*, u.username, u.email
        FROM Patients p
        JOIN Users u ON p.user_id = u.user_id
        ORDER BY p.city
    ''')

    patients = cursor.fetchall()
    cursor.close()

    return render_template('admin_patients_list.html', patients=patients)


@app.route('/admin/donations-list')
@login_required
@role_required('admin')
def admin_donations_list():
    cursor = mysql.connection.cursor()
    cursor.execute('''
        SELECT d.*,
               d_user.username as donor_name, d_user.email as donor_email,
               p_user.username as patient_name, p_user.email as patient_email,
               br.blood_group, br.units_required, br.urgency_level,
               pat.hospital_name
        FROM Donations d
        JOIN Donors don ON d.donor_id = don.donor_id
        JOIN Users d_user ON don.user_id = d_user.user_id
        JOIN Patients pat ON d.patient_id = pat.patient_id
        JOIN Users p_user ON pat.user_id = p_user.user_id
        JOIN blood_request_needed br ON d.request_id = br.request_id
        ORDER BY d.donation_date DESC
    ''')

    donations = cursor.fetchall()
    cursor.close()

    return render_template('admin_donations_list.html', donations=donations)

# ------------------------
# DONOR ROUTES

# -------------------------


@app.route('/donor/dashboard')

@login_required

@role_required('donor')

def donor_dashboard():

    user_id = session['user_id']

    cursor = mysql.connection.cursor()


    cursor.execute('SELECT * FROM Donors WHERE user_id = %s', (user_id,))

    donor = cursor.fetchone()


    if not donor:

        cursor.close()

        return redirect(url_for('complete_donor_profile'))


    cursor.execute('''

        SELECT d.*, br.blood_group, br.units_required

        FROM Donations d

        LEFT JOIN blood_request_needed br ON d.request_id = br.request_id

        WHERE d.donor_id = %s
        ORDER BY d.donation_date DESC
    ''', (donor['donor_id'],))
    donations = cursor.fetchall()


    # Get matching patients (same city, same blood group needed) - AUTO NOTIFY
    cursor.execute('''
        SELECT p.*, u.username, u.email, p.contact_number, br.request_id,
br.units_required, br.urgency_level
        FROM Patients p
        JOIN Users u ON p.user_id = u.user_id
        JOIN blood_request_needed br ON p.patient_id = br.patient_id
        WHERE p.city = %s AND br.blood_group = %s AND br.status = 'Pending'
        ORDER BY br.urgency_level DESC, br.request_date DESC
    ''', (donor['city'], donor['blood_group']))


    matching_patients = cursor.fetchall()


    # Create auto-notifications for matching patients
    if matching_patients:
        for patient in matching_patients:
            cursor.execute('''
                SELECT * FROM Notifications
                WHERE user_id = %s AND message LIKE %s

            ''', (patient['user_id'], f"%{donor['blood_group']}%"))



            existing = cursor.fetchone()



            if not existing:

                message = f"Donor {donor['contact_number']} available for
{donor['blood_group']} in {donor['city']}"

                cursor.execute('''

                    INSERT INTO Notifications (user_id, message, status)

                    VALUES (%s, %s, 'Unread')

                ''', (patient['user_id'], message))

                mysql.connection.commit()



    cursor.close()

    return render_template('donor_dashboard.html', donor=donor,
donations=donations, matching_patients=matching_patients)



@app.route('/donor/approve/<int:request_id>', methods=['POST'])

@login_required

@role_required('donor')

def approve_donation(request_id):

    user_id = session['user_id']
```

```
  cursor = mysql.connection.cursor()


  try:

    # Get the donor's ID

    cursor.execute('SELECT donor_id FROM Donors WHERE user_id = %s',
(user_id,))

    donor = cursor.fetchone()

    donor_id = donor['donor_id']


    # Get the patient ID from the request

    cursor.execute('SELECT patient_id FROM blood_request_needed WHERE
request_id = %s', (request_id,))

    request_data = cursor.fetchone()

    patient_id = request_data['patient_id']


    # Create a donation record

    cursor.execute('''

      INSERT INTO Donations (donor_id, patient_id, request_id, donation_date,
status)

      VALUES (%s, %s, %s, NOW(), 'Approved')

    ''', (donor_id, patient_id, request_id))


    # Update the blood request status to 'Approved'
```

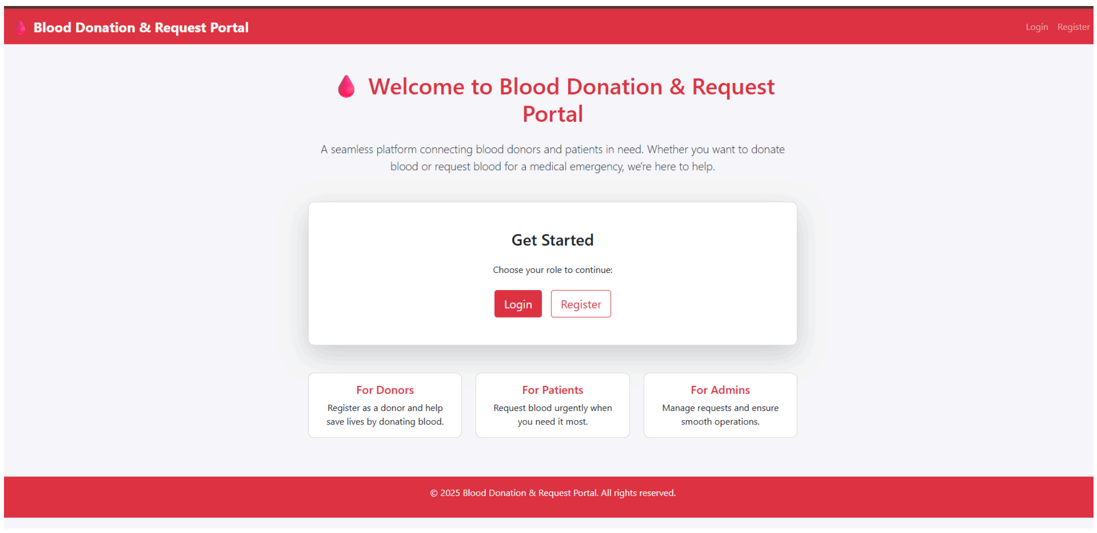
```
        cursor.execute('UPDATE blood_request_needed SET status = "Approved"
WHERE request_id = %s', (request_id,))


        mysql.connection.commit()

        cursor.close()


        flash('You have approved this donation! Thank you for helping!', 'success')

        return redirect(url_for('donor_dashboard'))


    except Exception as e:

        mysql.connection.rollback()

        cursor.close()

        flash(f'Error: {str(e)}', 'danger')

        return redirect(url_for('donor_dashboard'))


@app.route('/donor/complete-profile', methods=['GET', 'POST'])

@login_required

@role_required('donor')

def complete_donor_profile():

    if request.method == 'POST':

        try:

            user_id = session['user_id']
            blood_group = request.form.get('blood_group')

            age = request.form.get('age')

            gender = request.form.get('gender')

            city = request.form.get('city')

            state = request.form.get('state')

            contact_number = request.form.get('contact_number')



            # Debug: Print values

            print(f"DEBUG: Inserting donor - user_id={user_id},
blood_group={blood_group}, age={age}")



            cursor = mysql.connection.cursor()

            cursor.execute('''

                INSERT INTO Donors (user_id, blood_group, age, gender, city, state,

                contact_number, availability_status)

                VALUES (%s, %s, %s, %s, %s, %s, %s, %s)

            ''', (user_id, blood_group, age, gender, city, state, contact_number,
'Available'))



            mysql.connection.commit()

            cursor.close()



            print("DEBUG: Donor profile saved successfully!")
```

```
            flash('Profile completed successfully!', 'success')
            return redirect(url_for('donor_dashboard'))

        except Exception as e:
            print(f"DEBUG: Error saving donor profile: {str(e)}")
            flash(f'Error: {str(e)}', 'danger')
            return redirect(url_for('complete_donor_profile'))

    return render_template('complete_donor_profile.html')

# -------------------------
# PATIENT ROUTES
# -------------------------

@app.route('/patient/dashboard')
@login_required
@role_required('patient')
def patient_dashboard():
    user_id = session['user_id']
    cursor = mysql.connection.cursor()

    cursor.execute('SELECT * FROM Patients WHERE user_id = %s', (user_id,))
    patient = cursor.fetchone()


    if not patient:

        cursor.close()

        return redirect(url_for('complete_patient_profile'))


    cursor.execute('''

        SELECT * FROM blood_request_needed

        WHERE patient_id = %s

        ORDER BY request_date DESC

    ''', (patient['patient_id'],))

    requests = cursor.fetchall()


    # Get matching donors (same city, same blood group available) - AUTO NOTIFY

    cursor.execute('''

        SELECT d.*, u.username, u.email, d.contact_number

        FROM Donors d

        JOIN Users u ON d.user_id = u.user_id

        WHERE d.city = %s AND d.blood_group = %s AND d.availability_status =
'Available'

        ORDER BY d.age ASC

    ''', (patient['city'], patient['blood_group_needed']))
    matching_donors = cursor.fetchall()


    # Create auto-notifications for matching donors

    if matching_donors:

        cursor.execute('''

            SELECT * FROM blood_request_needed WHERE patient_id = %s AND status
= 'Pending'

        ''', (patient['patient_id'],))

        pending_request = cursor.fetchone()


        if pending_request:

            for donor in matching_donors:

                cursor.execute('''

                    SELECT * FROM Notifications

                    WHERE user_id = %s AND message LIKE %s

                ''', (donor['user_id'], f"%{patient['blood_group_needed']}%"))


                existing = cursor.fetchone()


                if not existing:

                    message = f"Patient {patient['contact_number']} needs
{patient['blood_group_needed']} at {patient['hospital_name']} ({patient['city']})"
```

```
            cursor.execute('''
                INSERT INTO Notifications (user_id, message, status)
                VALUES (%s, %s, 'Unread')
            ''', (donor['user_id'], message))
            mysql.connection.commit()

    cursor.close()
    return render_template('patient_dashboard.html', patient=patient,
requests=requests, matching_donors=matching_donors)


@app.route('/patient/complete-profile', methods=['GET', 'POST'])
@login_required
@role_required('patient')
def complete_patient_profile():
    if request.method == 'POST':
        user_id = session['user_id']
        blood_group_needed = request.form['blood_group_needed']
        hospital_name = request.form['hospital_name']
        city = request.form['city']
        state = request.form['state']
        contact_number = request.form['contact_number']
        cursor = mysql.connection.cursor()

        cursor.execute('''

            INSERT INTO Patients (user_id, blood_group_needed, hospital_name, city,
state, contact_number)

            VALUES (%s, %s, %s, %s, %s, %s)

        ''', (user_id, blood_group_needed, hospital_name, city, state,
contact_number))

        mysql.connection.commit()

        cursor.close()



        flash('Profile completed successfully!', 'success')

        return redirect(url_for('patient_dashboard'))



    return render_template('complete_patient_profile.html')



@app.route('/patient/create-request', methods=['GET', 'POST'])

@login_required

@role_required('patient')

def create_blood_request():

    user_id = session['user_id']

    cursor = mysql.connection.cursor()

    cursor.execute('SELECT * FROM Patients WHERE user_id = %s', (user_id,))

    patient = cursor.fetchone()
    if not patient:

        flash('Please complete your profile first!', 'warning')

        cursor.close()

        return redirect(url_for('complete_patient_profile'))



    if request.method == 'POST':

        blood_group = request.form['blood_group']

        units_required = request.form['units_required']

        urgency_level = request.form['urgency_level']



        cursor.execute('''

            INSERT INTO blood_request_needed (patient_id, blood_group,
units_required,

            urgency_level, status, request_date)

            VALUES (%s, %s, %s, %s, 'Pending', NOW())

        ''', (patient['patient_id'], blood_group, units_required, urgency_level))

        mysql.connection.commit()

        cursor.close()



        flash('Blood request created successfully!', 'success')

        return redirect(url_for('patient_dashboard'))
    cursor.close()

    return render_template('create_blood_request.html')


# ------------------------

# NOTIFICATIONS ROUTES

# ------------------------


@app.route('/notifications')

@login_required

def view_notifications():

    user_id = session['user_id']

    cursor = mysql.connection.cursor()


 cursor.execute('''

        SELECT * FROM Notifications

        WHERE user_id = %s

        ORDER BY notification_id DESC

    ''', (user_id,))


    notifications = cursor.fetchall()

    cursor.close()
    return render_template('notifications.html', notifications=notifications)


@app.route('/notification/read/<int:notif_id>')
@login_required
def mark_notification_read(notif_id):
    cursor = mysql.connection.cursor()
    cursor.execute('''
        UPDATE Notifications SET status = 'Read' WHERE notification_id = %s
    ''', (notif_id,))
    mysql.connection.commit()
    cursor.close()
    return redirect(url_for('view_notifications'))
if __name__ == '__main__':
    app.run(debug=True)
```

CHAPTER 5

SCREEN SHOTS



Welcome to Blood Donation & Request Portal

A seamless platform connecting blood donors and patients in need. Whether you want to donate blood or request blood for a medical emergency, we're here to help.

Get Started

Choose your role to continue:

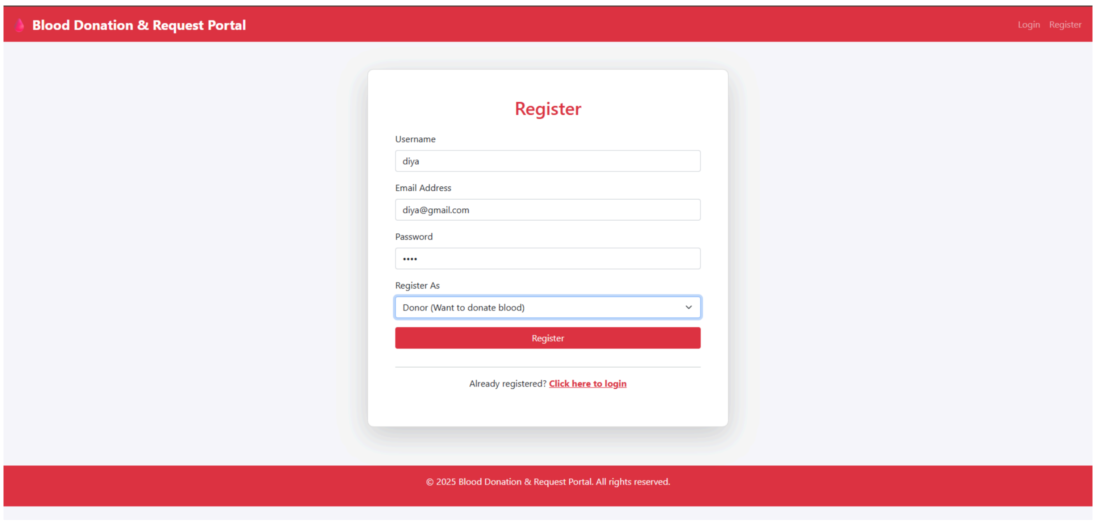
For Donors
Register as a donor and help save lives by donating blood.

For Patients
Request blood urgently when you need it most.

For Admins
Manage requests and ensure smooth operations.

© 2025 Blood Donation & Request Portal. All rights reserved.

Fig 5.1 Introduction page



Register

Username: diya

Email Address: diya@gmail.com

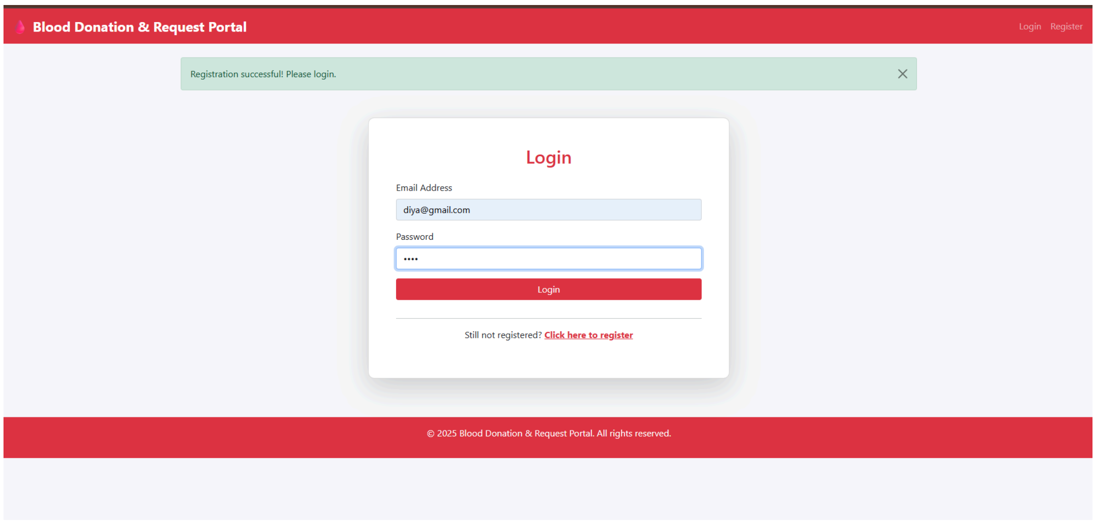
Password: ••••

Register As: Donor (Want to donate blood)

Already registered? Click here to login

© 2025 Blood Donation & Request Portal. All rights reserved.

Fig 5.2 Register page


Blood Donation & Request Portal

Login Register

Registration successful! Please login.

Login

Email Address

diya@gmail.com

Password

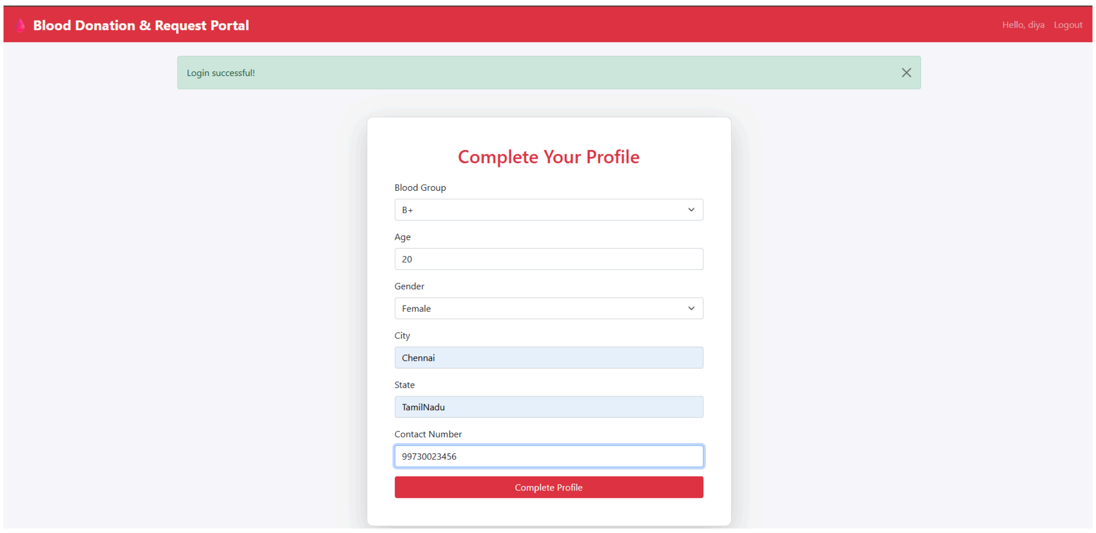
••••

Login

Still not registered? Click here to register

© 2025 Blood Donation & Request Portal. All rights reserved.

Fig 5.3 Login page


Blood Donation & Request Portal

Hello, diya Logout

Login successful!

Complete Your Profile

Blood Group

B+

Age

20

Gender

Female

City

Chennai

State

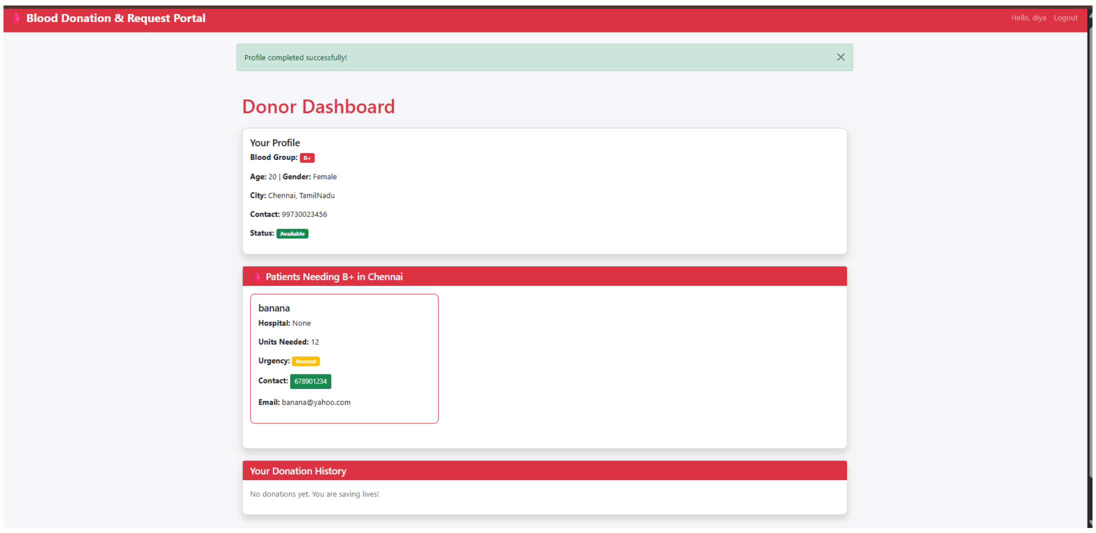
TamilNadu

Contact Number

99730023456

Complete Profile

Fig 5.4 Profile Completion



Blood Donation & Request Portal

Hello, diya Logout

Profile completed successfully!

Donor Dashboard

Your Profile

Blood Group: B+

Age: 20 | **Gender:** Female

City: Chennai, TamilNadu

Contact: 99730023456

Status: Available

Patients Needing B+ in Chennai

banana

Hospital: None

Units Needed: 12

Urgency: Normal

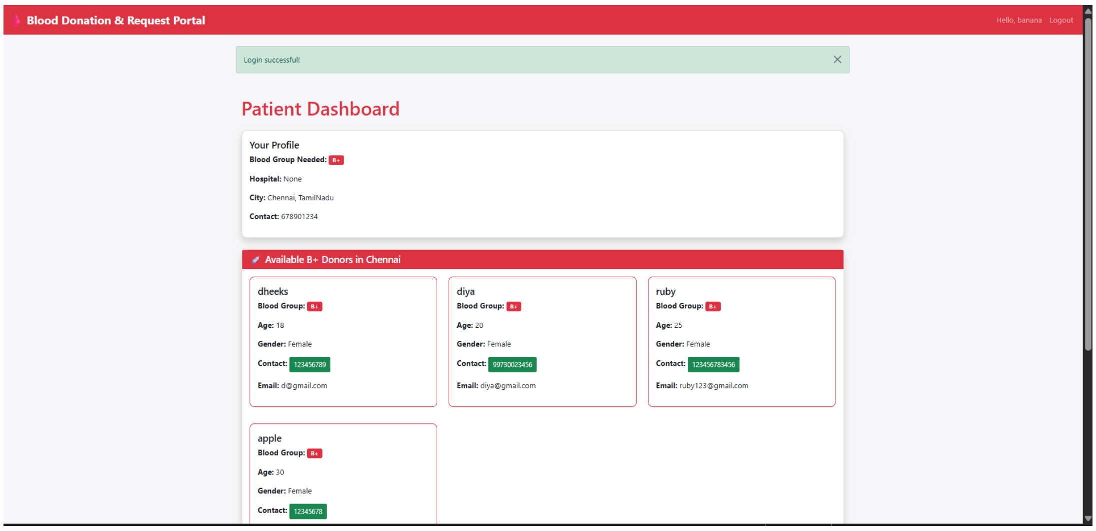
Contact: 678901234

Email: banana@yahoo.com

Your Donation History

No donations yet. You are saving lives!

Fig 5.5 Donor Dashboard



Blood Donation & Request Portal

Hello, banana Logout

Login successful!

Patient Dashboard

Your Profile

Blood Group Needed: B+

Hospital: None

City: Chennai, TamilNadu

Contact: 678901234

Available B+ Donors in Chennai

dheeks

Blood Group: B+

Age: 18

Gender: Female

Contact: 123456789

Email: d@gmail.com

diya

Blood Group: B+

Age: 20

Gender: Female

Contact: 99730023456

Email: diya@gmail.com

ruby

Blood Group: B+

Age: 25

Gender: Female

Contact: 123456783456

Email: ruby123@gmail.com

apple

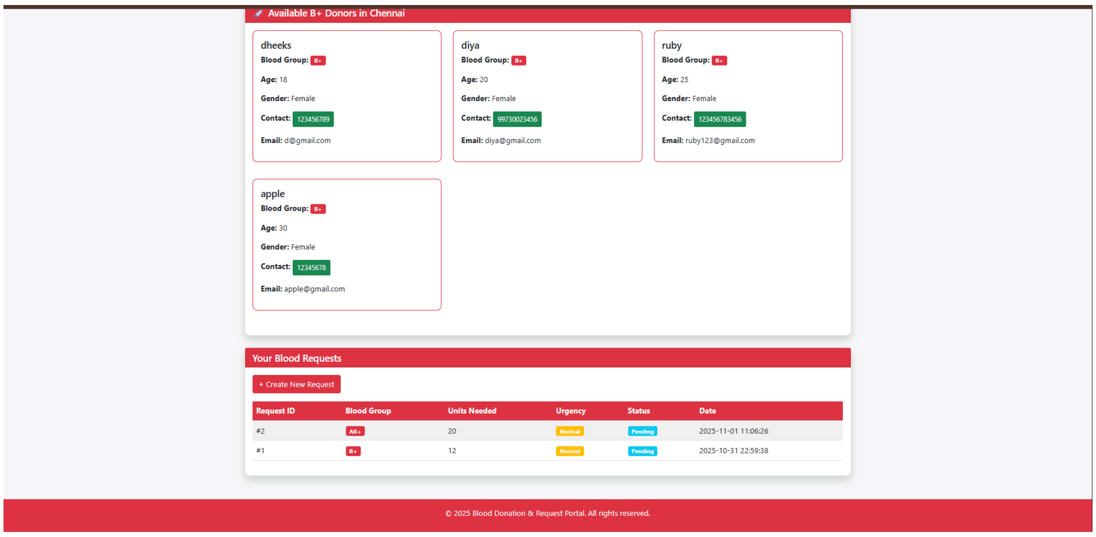
Blood Group: B+

Age: 30

Gender: Female

Contact: 12345678

Fig 5.6 Patient Dashboard(a)



Available B+ Donors in Chennai

dheeks
Blood Group: B+
Age: 18
Gender: Female
Contact: 123456789
Email: d@gmail.com

diya
Blood Group: B+
Age: 20
Gender: Female
Contact: 99730023456
Email: diya@gmail.com

ruby
Blood Group: B+
Age: 25
Gender: Female
Contact: 123456783456
Email: ruby123@gmail.com

apple
Blood Group: B+
Age: 30
Gender: Female
Contact: 12345678
Email: apple@gmail.com

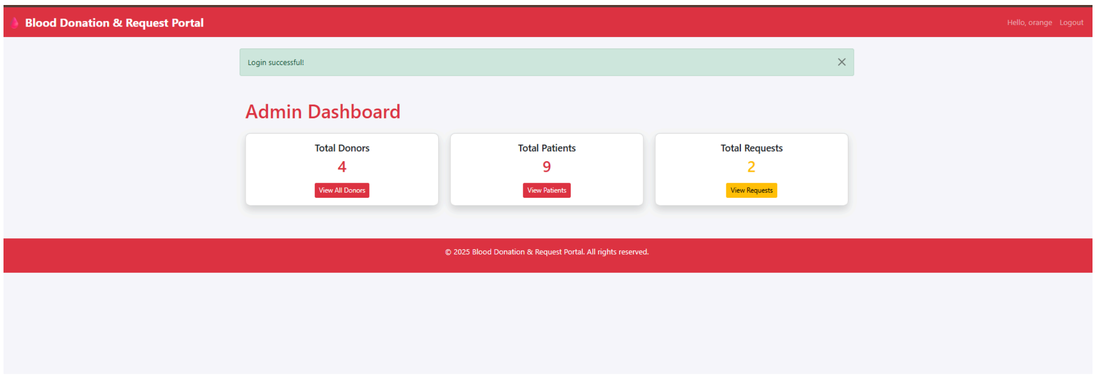
Your Blood Requests

+ Create New Request

Request ID	Blood Group	Units Needed	Urgency	Status	Date
#2	AB+	20	Normal	Pending	2025-11-01 11:06:26
#1	B+	12	Normal	Pending	2025-10-31 22:59:38

© 2025 Blood Donation & Request Portal. All rights reserved.

Fig 5.6(b) Patient Dashboard



Blood Donation & Request Portal

Hello, orange Logout

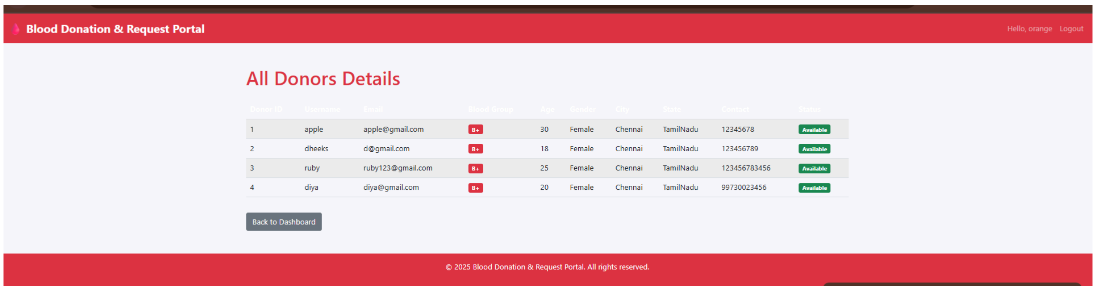
Login successfull!

Admin Dashboard

Total Donors	Total Patients	Total Requests
4	9	2
View All Donors	View Patients	View Requests

© 2025 Blood Donation & Request Portal. All rights reserved.

Fig 5.7 Admin Dashboard(a)



Blood Donation & Request Portal

Hello, orange Logout

All Donors Details

Donor ID	Username	Email	Blood Group	Age	Gender	City	State	Contact	Status
1	apple	apple@gmail.com	B+	30	Female	Chennai	TamilNadu	12345678	Available
2	dheeks	d@gmail.com	B+	18	Female	Chennai	TamilNadu	123456789	Available
3	ruby	ruby123@gmail.com	B+	25	Female	Chennai	TamilNadu	123456783456	Available
4	diya	diya@gmail.com	B+	20	Female	Chennai	TamilNadu	99730023456	Available

Back to Dashboard

© 2025 Blood Donation & Request Portal. All rights reserved.

Fig 5.7 Admin Dashboard(b)

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

In this project, a secure and efficient Blood Donation and Request Portal has been developed to simplify and manage the process of blood donation and requests. The system enables users to register, search for suitable donors, and make blood requests, while administrators can manage donor data, verify user details, and monitor overall activities. It provides a centralized and reliable platform that improves coordination between donors, recipients, and hospitals, ensuring timely blood availability during emergencies.

In the future, this system can be enhanced by integrating AI-based donor-recipient matching, automated SMS and email alerts, mobile app integration for real-time updates, and GPS tracking to locate nearby donors quickly. Features such as blood donation camps scheduling, integration with hospital management systems, real-time blood stock monitoring, and data analytics dashboards for tracking donation trends can further improve efficiency. Additionally, blockchain technology can be implemented for secure and transparent record-keeping, ensuring complete data integrity and trust across all users.

REFERENCES

1. https://www.w3schools.com/sql/
2. https://www.tutorialspoint.com/sqlite/index.htm
3. https://www.wikipedia.org/
4. https://www.learnpython.org/
5. https://www.codecademy.com/learn/learn-python