1.INTRODUCTION

i. INTRODUCTION

The **Blood Bank Management System** is a web-based application designed to streamline the process of blood donation, blood requests, and inventory management in blood banks. It serves as a centralized platform that connects blood donors, recipients, and administrators, ensuring efficient handling of blood availability and related data.

Blood plays a vital role in saving lives during emergencies such as surgeries, accidents, and medical conditions like anaemia or cancer. However, traditional methods of managing blood donations and requests often lead to delays, miscommunication, and data inaccuracy. This system aims to address those challenges by digitizing the entire process, making it more transparent, accessible, and manageable.

The system allows donor**s** to register and provide their blood group and availability, while user**s** in need of blood can request specific types and quantities. The administrator module handles approval of requests, updates the blood inventory, and monitors donor and recipient records. This not only helps maintain accurate records of blood units available but also reduces the risk of blood shortages and improves overall healthcare services.

a. Project Title

BLOOD BANK MANAGEMENT SYSTEM

b. Category

Web-based application.

c. Overview

The **Blood Bank Management System** is a comprehensive web application developed to automate and manage the various operations of a blood bank. The system provides a digital platform where blood donors, recipients, and administrators can interact efficiently to fulfill blood requirements in a timely manner.

This system is primarily divided into three main modules:

User Module – Allows users to register either as a donor or a recipient. Donors can enter their blood group and availability, while recipients can submit blood requests based on their needs.

Admin Module – Enables the admin to manage donor and recipient information, approve or deny blood requests, monitor available blood units, and update the inventory accordingly.

Inventory Management – Tracks the availability of different blood groups and updates the stock as donations are made or blood units are issued.

.

ii. BACKGROUND

Blood is a crucial component in medical treatments and emergency situations. Hospitals and healthcare centres often rely on blood banks to supply various blood groups for surgeries, trauma care, and treatment of diseases such as cancer, anaemia, and bleeding disorders. However, managing blood donations and requests manually can lead to several issues, including data loss, delays, miscommunication, and difficulty in maintaining accurate inventory records.

Traditionally, blood banks have used paper-based systems or outdated software to manage donor and recipient information. These methods are inefficient, time-consuming, and prone to human error. Moreover, during emergencies, the inability to quickly access information about available blood types can cost lives.

With the advancement of information technology, there is a growing need for a robust and automated solution to manage blood bank operations. A **Blood Bank Management System** provides a centralized platform to store and manage donor and recipient data, track blood inventory, and facilitate smooth communication between donors, recipients, and administrators.

This project is designed to address these issues by developing a digital system using **HTML, CSS, PHP, and MySQL**. It simplifies blood donation and request processes, ensures real-time inventory updates, and promotes transparency and efficiency in managing blood bank services

a. Brief note on the existing System

Our website is user-friendly and convenient for customers to use. We have the following:

A Home page which navigates to Donate or request by register/login, about us section, Feedback section, contact info section which provides further details about our website and option called Online where user can directly keep in touch with donor who is ready to donate It also consists of home delivery and payment section.

➢ **About us page:**

This page consists of details about our shop and services we provide online. It also includes the information about range of products. Our team is dedicated to sourcing high quality items and ensuring customer's shopping experience is convenient and enjoyable.

This page has navigation for other pages in this website.

➢ **Feedback Page:**

In the **Blood Bank Management System**, the Feedback Form is an essential feature that allows users, donors, or recipients to share their thoughts, suggestions, or issues they faced while using the system. This helps administrators collect user feedback to improve the platform's performance, user experience, and reliability.

➢ **Online**

This is our new concept page which is a feature for users where if there is any donor who are interested in donating blood .can register here no login is required. In any emergency case if there is a need for blood user can search for required blood and can contact donors directly the contact information of donor will be shown up, user can directly contact those donors.

➢ **Contact page:**

- On this page, we provide our Contact details such as number.
- We also provided our website links and Email for further queries.
- Additionally, we have provided emergency helpline numbers, contact number etc.

➢ **Login Page:**

The Login and Register pages are the entry points for users (donors or recipients) to access the system. They serve three main purposes:

User Authentication – Ensures that only registered users can donate, request blood, or view their data.
Data Security – Protects sensitive data like donor details and blood requests.
Personalized Access – After login, users see only their information (e.g., request status, donation history).

iii. OBJECTIVES OF OUR SYSTEM

The main objective of the **Blood Bank Management System** is to develop a secure, efficient, and user-friendly web application that automates and simplifies the process of blood donation, blood requests, and blood inventory management. The system aims to bridge the gap between blood donors and recipients while ensuring transparency, quick access, and accurate record-keeping in blood bank operations.

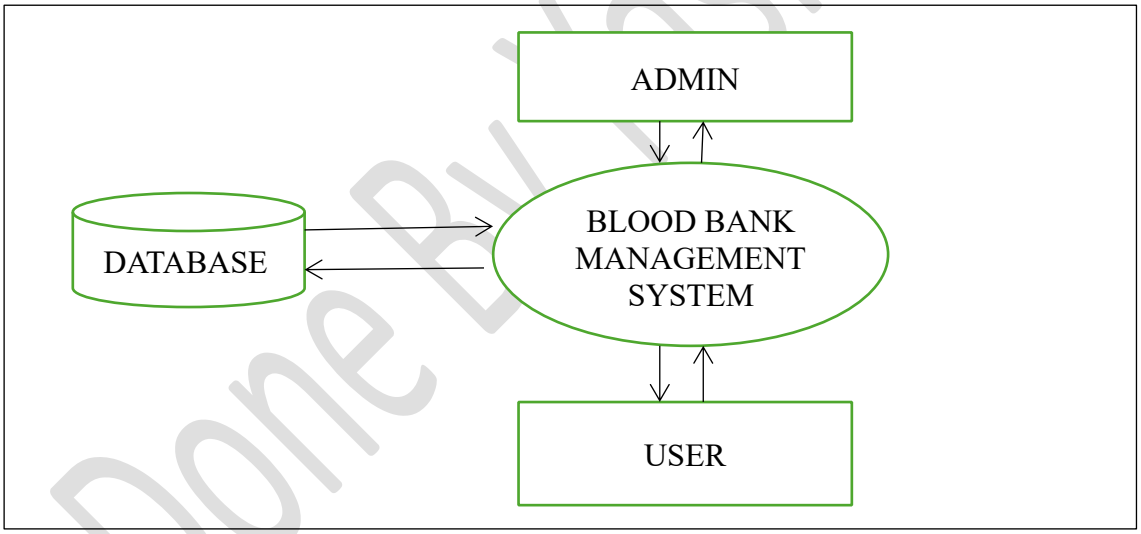
iv. SCOPE OF THE SYSTEM

The system supports online registration and login for both blood donors and recipients.

- Allows donors to submit blood donation details, including blood group, availability,and contact info.
- Enables recipients to request blood by selecting blood group and entering request details.
- Provides an admin dashboard to:
    1. View, approve, or deny blood requests
    2. Manage donor and recipient information
    3. Update and monitor blood stock levels
- Maintains a real-time blood inventory categorized by blood groups (A, B, AB, O, etc.).
- Ensures data accuracy and security through validation and login-based access.
- Accessible to users through a web browser, with a user-friendly and responsive interface.
- Designed to handle multiple users simultaneously, ensuring smooth interaction.
- Reduces manual paperwork and eliminates human errors in tracking blood units and request status.
- Helps in emergency situations by quickly identifying available blood types and notifying admins

v. SYSTEM ARCHITECTURE

Architecture design is the process of decomposing a large complex system into small subsystem. These subsystems are meant for providing some related services. The architecture design is basically a layout or a framework of the system for the subsystem control or communication.



a. Admin Role:

In an blood bank management system, the admin role typically involves managing and maintaining the system's functionality, ensuring smooth operations, and overseeing various administrative tasks.

b. User Role:

A Donor can register, log in, and submit their blood donation details, as well as view their donation history. A Recipient (or requester) can sign up, request blood, and track the status of their request. Each role has limited access based on its purpose to ensure security and proper data handling.

vi. SOFTWARE AND HARDWARE USED

a. **Software Used**

1. Web Technology: PHP 5.6
2. Web Components: HTML5/CSS/JavaScript
3. Software's: XAMPP, NotePad++
4. Database (Backend): MYSQL Server
5. Web Server: Apache

b. **Hardware Used**

1. Processor: Intel core i3
2. Processor Speed: Minimum 2 GHz
3. RAM: 4GB of RAM or above
4. Hard Disk: Minimum 40GB free space

Input Device: Mouse, Keyboard

2. SOFTWARE REQUIREMENTS AND SPECIFICATIONS

i. SOFTWARE REQUIREMENTS AND SPECIFICATIONS

SRS stands for Software Requirement Specification, which is a document that fully describes the expected behaviour of a software system. Functional requirements are documented in an SRS, as are non- functional requirements such as performance goals and description of quality attributes. A Software Requirement Specification is a description of a software system to be developed. The SRS document intended to give a complete overview of online marketplace. This also gives the complete development and performance of the system.

OVERALL DESCRIPTION

Product Perspective

The Blood Bank Management System offers a streamlined and efficient approach to managing the critical operations of blood donation and distribution. It enhances the coordination between donors, recipients, and hospitals by maintaining accurate and real-time records of blood availability, donor details, and request statuses. This system helps minimize manual errors, reduces processing time, and ensures timely access to life-saving blood supplies. By digitizing records and automating processes, it not only improves transparency and traceability but also promotes better inventory management and decision-making during emergencies.

Product Functions

This web application performs the following functions:

1. It allows the admin to log in and log out.
2. It allows the user to login and register data .
3. It allows the user to search for data.
4. It allows the user to report.
5. It allows the admin to manage and control the data.

a. User Characteristics

This program keeps accurate records of donor information, blood types, and donation history. It also includes an effective inventory management system that tracks the availability of various blood groups and components in real-time. The system handles blood requests from hospitals or patients, ensuring quick and accurate processing.

b. General Constraints

1. Need any browser to view the application.
2. User age must be above 18.
3. Technical knowledge.

Product Functions

This web application performs the following functions:

4. Privacy concern.

c. Software Requirements

Operating System: Windows 7 & above
Web Server: Xampp, Code Editor
Language: PHP Web Browser: Google Chrome/Internet Explorer
Database: MySQL and phpMyAdmin
IDE: Notepad++

ii. FUNCTIONAL REQUIREMENTS

There are two modules in the project.

1. User
2. Admin

a. **User Module**

1. User
2. Admin

a. **User Module**

Users can log in securely using their credentials. After logging in, users can access features like searching for available blood types, viewing donor information, and checking blood stock availability in different locations. This module ensures that users have a smooth and secure experience while interacting with the system, enabling quick access to important information related to blood donation and requests.

b. **Admin Module**

Login: The admin can login to his account using his username and password.
Admins are responsible for managing and maintaining the overall functionality of the system. Admins have the authority to log in securely and access all system data. They can add, update, or delete donor and patient records, manage blood inventory, and handle blood request approvals. This module also allows admins to monitor blood stock levels, generate reports, and ensure that all information is accurate and up to date. By having full control over the system's data and operations,

searching for available blood types, viewing donor information, and checking blood stock availability in different locations. This module ensures that users have a smooth and secure experience while interacting with the system, enabling quick access to important information related to blood donation and requests.

b. Admin Module

Login: The admin can login to his account using his username and password.
Admins are responsible for managing and maintaining the overall functionality of the system. Admins have the authority to log in securely and access all system data. They can add, update, or delete donor and patient records, manage blood inventory, and handle blood request approvals. This module also allows admins to monitor blood stock levels, generate reports, and ensure that all information is accurate and up to date. By having full control over the system's data and operations,

DESIGN

i. SYSTEM DESIGN

a. INTRODUCTION

a. INTRODUCTION

System design is the architecture built on the basis of the system analysis. The purpose of the the design phase is to plan a solution for the problem specified by the requirements. The system design describes the data to be input, calculated and stored. It is built on the required input and output design of the files. It aims to identify the module that should be present in the system the specification of this module and how they interact with each other to produce the desired result. The design activity is often divided into separate phase system design is a sometimes also called top level design. At the end of the system design all the major data structure file format and the major modules in the system and their specification are decided.

ii. DESCRIPTION OF FUNCTION WITH DECOMPOSITION

a. **ADMIN SIDE**:

. Functional Component 1: Login

1. Input: username, password.
2. Process: The system checks for the correctness of the username and password
3. Output: Admin will be taken to the dashboard page on successful login

. Functional Component 2: Manage User Request & Donation

1. Input: id, name, age, blood types, unit, disease, email, phone, and address.
2. Process: The system will manage to approve, pending, deny, and delete the requests.
3. Output: Admin successfully adds the data, views the details, deny the request, successfully deleted the request and its details.

. **Functional Component 3: Manage users**

1. Input id, name, age, blood types, unit, disease, email, phone, and address.
2. Process: The system will manage to edit and delete the users and their details.
3. Output: Admin successfully edited the details of the users or admin successfully deleted the user and their details.

b . User Registration

design phase is to plan a solution for the problem specified by the requirements. The system design describes the data to be input, calculated and stored. It is built on the required input and output design of the files. It aims to identify the module that should be present in the system the specification of this module and how they interact with each other to produce the desired result. The design activity is often divided into separate phase system design is a sometimes also called top level design. At the end of the system design all the major data structure file format and the major modules in the system and their specification are decided.

ii. DESCRIPTION OF FUNCTION WITH DECOMPOSITION

a. **ADMIN SIDE**:

. Functional Component 1: Login

1. Input: username, password.

2. Process: The system checks for the correctness of the username and password

3.Output: Admin will be taken to the dashboard page on successful login

. Functional Component 2: Manage User Request & Donation

1. Input: id, name, age, blood types, unit, disease, email, phone, and address.

2. Process: The system will manage to approve, pending, deny, and delete the requests.

3. Output: Admin successfully adds the data, views the details, deny the request, successfully deleted the request and its details.

. **Functional Component 3: Manage users**

1. Input id, name, age, blood types, unit, disease, email, phone, and address.

2. Process: The system will manage to edit and delete the users and their details.

3. Output: Admin successfully edited the details of the users or admin successfully deleted the user and their details. **b . User Registration**

- Input: Name, Email, Password, Phone, Blood Group, Role (Donor or Recipient)
- Process: Validate form data, Hash password Store user data in the database
- Output: Registration successful, message Redirect to login page

c User Login

- Input:
  - Email and Password
- Process:
  - Check user credentials against database
  - Verify password
  - Start user session
- Output:
  - Redirect to user dashboard (Donor or Recipient)
  - Error message if login fails

d Blood Donation (Donor)

- Input: Blood group, date of donation, quantity, contact info
- Process: Validate input, Store donation details in database, Update blood inventory
- Output: Confirmation message. Display donation history

e Blood Request (Recipient)

Input: Blood group, quantity needed, reason, contact info
Process: Validate request details, Store request in database, Notify admin for approval
Output: Request submitted message, Show request status (Pending/Approved/Denied)

f View Request or Donation Status

Input: User ID (session-based)

Process: Fetch request or donation records from database

Output: Display list of previous requests or donations with status

iii DESCRIPTION OF FUNCTION

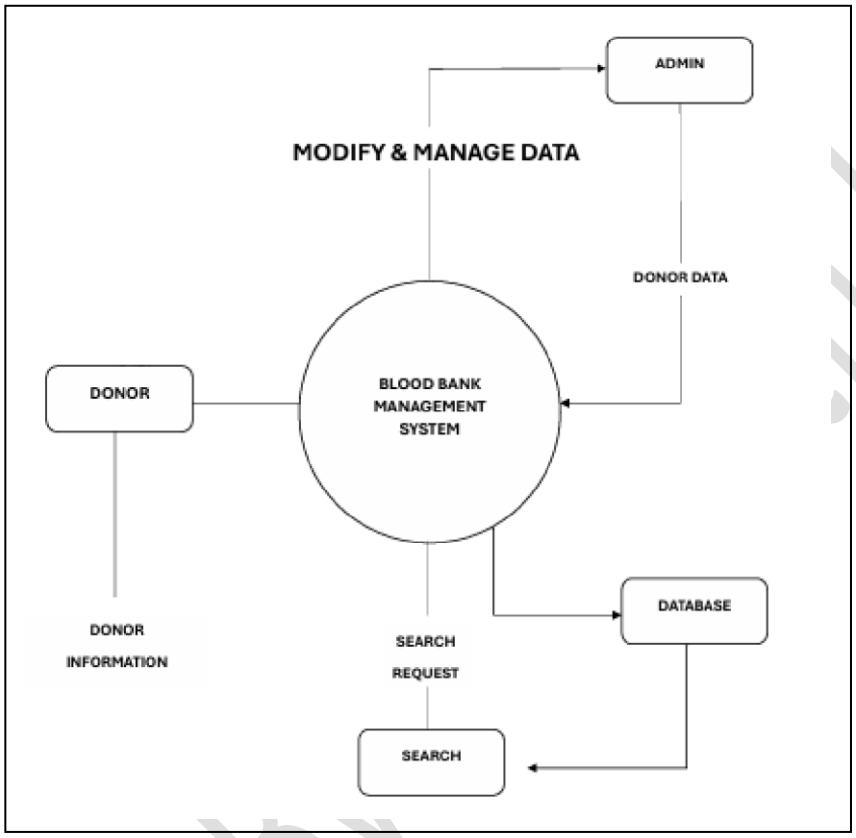
iii DESCRIPTION OF FUNCTION

Context Flow Diagram (CFD)

Context Flow Diagram (CFD)

Context Flow Diagram is a top level (also known as level 0) data flow datagram. It contains only one process node that generalizes the functions of the entire system in relationship to external entities. In context diagram the entire system is treated as single process and all its inputs, outputs, sinks, and sources are identified.

Diagram



The above diagram is context flow diagram. It consists of two attributes they are: Admin and User. Admin manages all the entities of the Project.

b Data Flow Diagrams (DFD)

1. A Data Flow Diagram is a graphical representation of a system or a portion of the system. It consists of data flows, process, sources and sink and stores all the description through the use of easily understandable symbols.

2. DFD is one of the most important modelling tools. It is used to model the system, components that interact with the system, uses the data and information flows in the system.

3. DFD shows the information moves through the and how it is modified by a series of transformations. It is a graphical technique that depicts information moves from input or output.

4. DFD is also known as bubble chart or Data Flow Graphs. DFD's may partition into a level that represents increasing information flows and functional details.

Rules Regarding DFD Construction:

1. A process cannot have only outputs.

2. A process cannot have only inputs.

3. The inputs to a process must be sufficient to produce the outputs from the process.

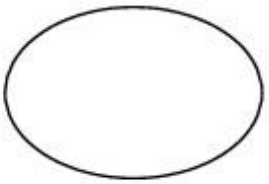
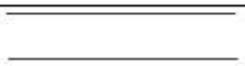
4. All data stores must be connected to at least one process.

5. All data stores must be connected to a source or sink.

6. A data flow can have only one direction off low multiple data flows to and/or from the same process and data store must be shown by separate arrows.

7. If the exact same data flows to two separate arrows it should be represented by a forked arrow.

8. Data can't flow directly back into the process it has just left. All data flows must be named using a noun phrase.

Name	Notation	Description
Process		A processor task performed bythe system.
Datastore		Datastores are repositories of data in the system.They are sometimes also referred to as files.
Dataflows		Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it.
External Entity		External entities are objects outside the system with which the system communicates. External Entities are sources and destinations of the system's inputs and outputs.

Here's an explanation of the updated Data Flow Diagram (DFD) for the Blood Bank Management System, including the new online blood search feature:

Entities:

1. Donor – Registers by submitting personal and medical data.

2. Admin – Manages donor information (update/delete).

3. User – A public user who can search for available blood.

Processes:

1. Donor Registration (implicit in this version):

• Donors provide details (blood group, name, etc.), which are sent to the Database.

2. Manage Donor Data (Admin):

• Admin retrieves donor data from the database.

• Admin updates or deletes donor records.

• Changes are written back to the database

.

3. Search Blood Availability (User):

• Public users submit a search query (like blood type or location).

• The system processes the request and retrieves matching donor info from the Database.

• The search results (filtered donor information) are displayed back to the user.

4. Data Store:

• Database: The central storage for all donor data.

• Donor submissions

• Admin updates

• Responds to user queries with filtered search results

Data Flows (Arrows):

• User → Search Blood Availability: Sends a search query

• Search Blood Availability → Database: Requests matching records

• Database → Search Blood Availability: Returns matching results

• Search Blood Availability → User: Displays search results

• Donor → Database: Submits donor details

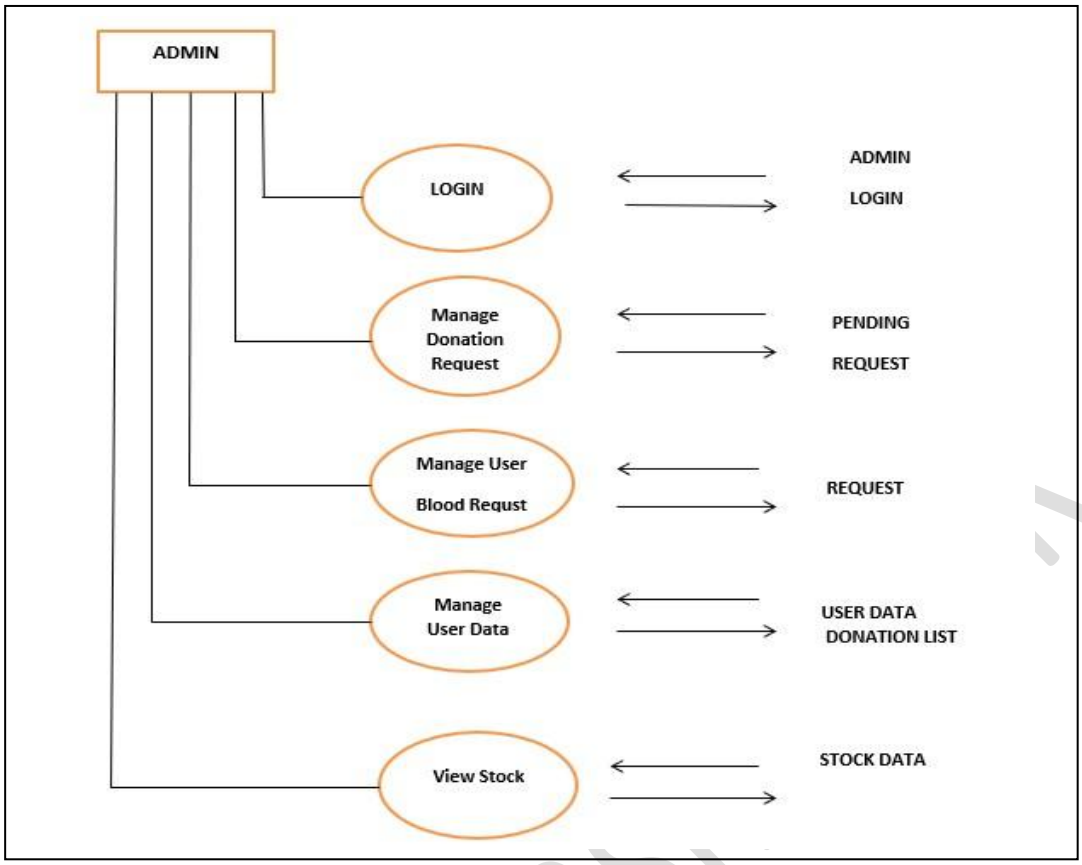
• Admin → Manage Donor Data: Views donor information

• Manage Donor Data → Database: Updates donor records

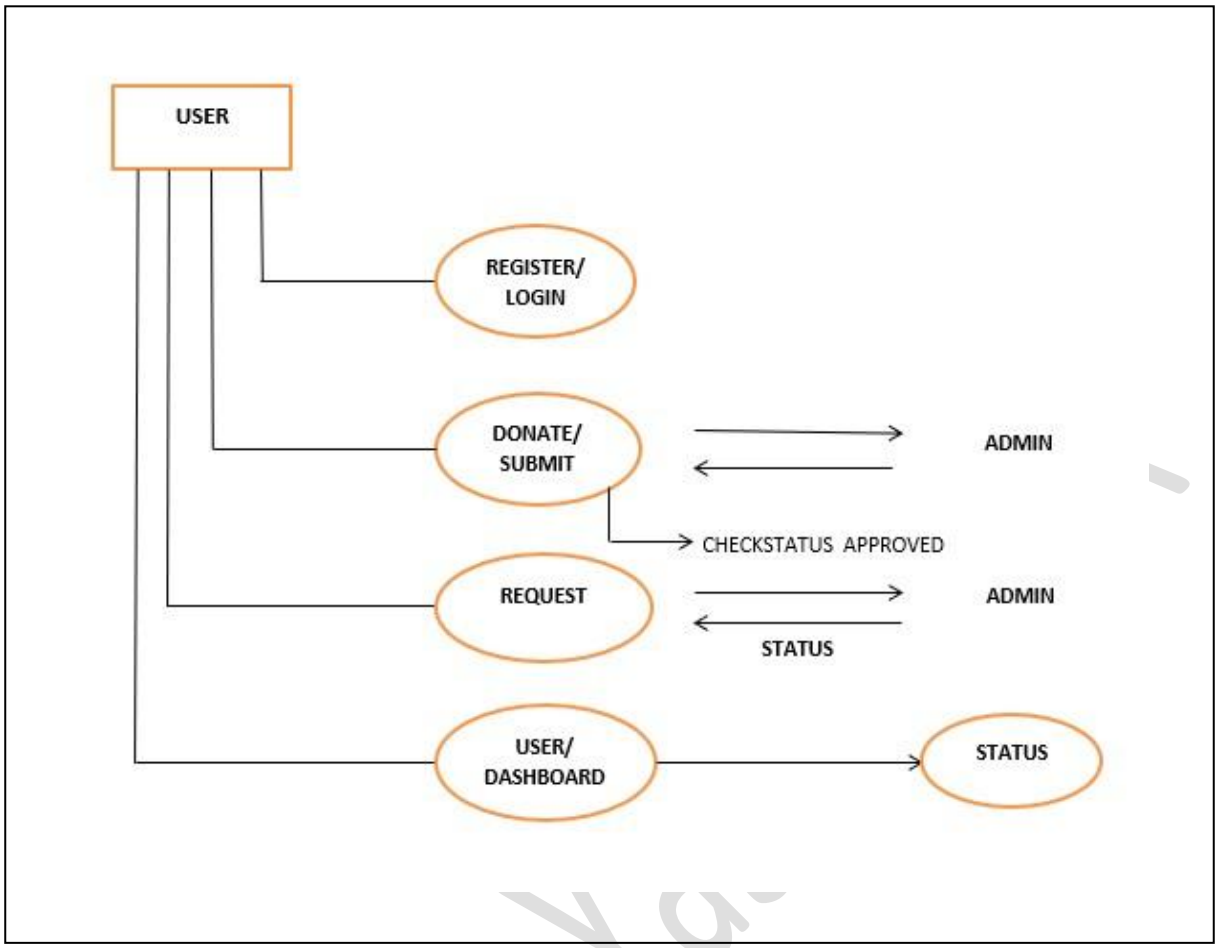
• Database → Manage Donor Data: Provides current donor data

This DFD now supports public search functionality alongside donor registration and admin data control.

Admin Interface


ADMIN
LOGIN
ADMIN
LOGIN
Manage
Donation
Request
PENDING
REQUEST
Manage User
Blood Requst
REQUEST
Manage
User Data
USER DATA
DONATION LIST
View Stock
STOCK DATA

User interface



iv DATABASE DIAGRAM

a. DATABASE SCHEMA

A database schema is the structure that defines the organization of data within a database. It provides a blueprint for how data is stored, organized, and managed. A schema specifies the tables, fields (columns), relationships, views, indexes, and constraints within a database.

In simple terms, it is the framework or architecture that outlines the arrangement of the database's components.

Components of a Database Schema:

1. **Tables:**

- These are the core components of the schema, representing entities or objects in the system (e.g., Donor, Blood_Request, Admin).
- Each table contains rows (records) and columns (fields or attributes).

2. **Columns (Fields/Attributes):**

- Each table consists of several columns, which define the type of data stored (e.g.,donor_id, name, blood_type).

• Columns can have various data types, such as VARCHAR, INT, DATE, etc.

3. Primary Keys:

• A primary key is a unique identifier for each record in a table (e.g., donor_id for the Donor table).

• It ensures that each record can be uniquely identified.

4. Foreign Keys:

• A foreign key is a field (or combination of fields) in a table that links to the primary key in another table, establishing a relationship between the two tables (e.g., donor_id in the Blood_Request table references donor_id in the Donor table).

5. Relationships:

• These define how tables are related to each other. Relationships can be:

• One-to-One: One record in a table is related to one record in another table.

• One-to-Many: One record in a table is related to multiple records in another table.

• Many-to-Many: Many records in one table are related to many records in another table (usually managed by a junction table).

V DATABASE DIAGRAM

DATABASE NAME

- **BLOOD_BANK**

TABLE NAMES

- **users**
- **pending_donations**
- **requests**
- **donations**
- **blood_stocks**

Table name and constraints.

users

users		
id	Int(11)	PRIMARY KEY
name	Varchar(100)	
email	Varchar(100)	UNIQUE
password	Varchar(255)	

pending_donations

pending_donations		
id	Int(11)	PRIMARY KEY
user_id	Int(11)	index
name	Varchar(100)	
age	Int(11)	
blood_type	Varchar(10)	
units	Int(11)	
diseases	Varchar(255)	
phone	Varchar(20)	
email	Varchar(100)	
address	Text	
status	Enum(pending,approved,denied)	
requested_at	Timestamp	

requests

requests		
id	Int(11)	PRIMARY KEY
user_id	Int(11)	INDEX
blood_type	Varchar(10)	
units	Int(11)	
status	Enum(pending,approved,denied)	
requested_at	Timestamp	

donations

donations		
id	Int(11)	PRIMARY KEY
name	Varchar(100)	
age	Int(11)	
blood_type	Varchar(10)	
units	Int(11)	
diseases	Varchar(255)	
phone	Varchar(20)	
email	Varchar(100)	
address	Text	
donated_at	Timestamp	

blood_stock

blood_stock		
blood_type	Varchar(10)	
units	Int(11)	

a.TABLE CREATING QUERY

```
Create the database
CREATE DATABASE IF NOT EXISTS blood_bank;
USE blood_bank;

-- Users table
CREATE TABLE users ( id INT
AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100), email
VARCHAR(100) UNIQUE,
  password VARCHAR(255)
);

-- Donations table
CREATE TABLE donations ( id INT
AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100),
  age INT, blood_type
VARCHAR(10),
  units INT, disease
VARCHAR(255), phone
VARCHAR(20), email
VARCHAR(100), address
TEXT,
  donated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Requests table
CREATE TABLE requests ( id INT
AUTO_INCREMENT PRIMARY KEY, user_id
INT,
  blood_type VARCHAR(10),
  units INT,
  status ENUM('pending', 'approved', 'denied') DEFAULT 'pending',
  requested_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id)
);

-- Blood stock table
CREATE TABLE blood_stock (
  blood_type VARCHAR(10) PRIMARY KEY,
  units INT DEFAULT 0
);

CREATE TABLE pending_donations ( id INT
AUTO_INCREMENT PRIMARY KEY, user_id
INT,
  name VARCHAR(100),
  age INT,
  blood_type VARCHAR(10),
  units INT,
```

```
    disease VARCHAR(255),     phone VARCHAR(20),     email
VARCHAR(100),     address TEXT,     status ENUM('pending',
'approved', 'denied') DEFAULT 'pending',
    requested_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

vi ER DIAGRAM

Entity relationship diagram is used in modern database software. Software engineering is to illustrate logical structure of database. It is a relational schema database, modelling method, used to model a system and approach. This approach is commonly used in database design. The diagram created using this are called entity relationship diagram .The ER diagram depicts the various relationships among entities, considering each object as an entity. Relationship depicts the relationship between data objects. The ERD is the notation that is used to conduct the data modelling activity.

- Entity

Entity is a thing, which we want to store information. It is an elementary basic building block of storing information about business process. An entity represents an object desired within the information system about which you want to store information.

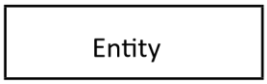
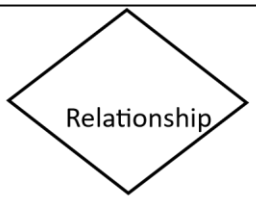
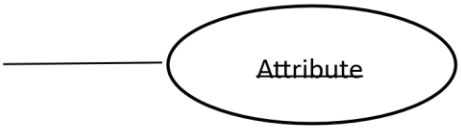
- Relationship

A relationship is a named connection, associated between entities, or used to relate two or more entities with the same common attributes or meaningful interaction between the object.

1. Many-to-many
2. Many-to-one
3. One-to-many
4. One-to-one

Attributes

Attributes are the properties of entities and relationship, description of the entity. Attributes are elementary pieces of information attached to an entity

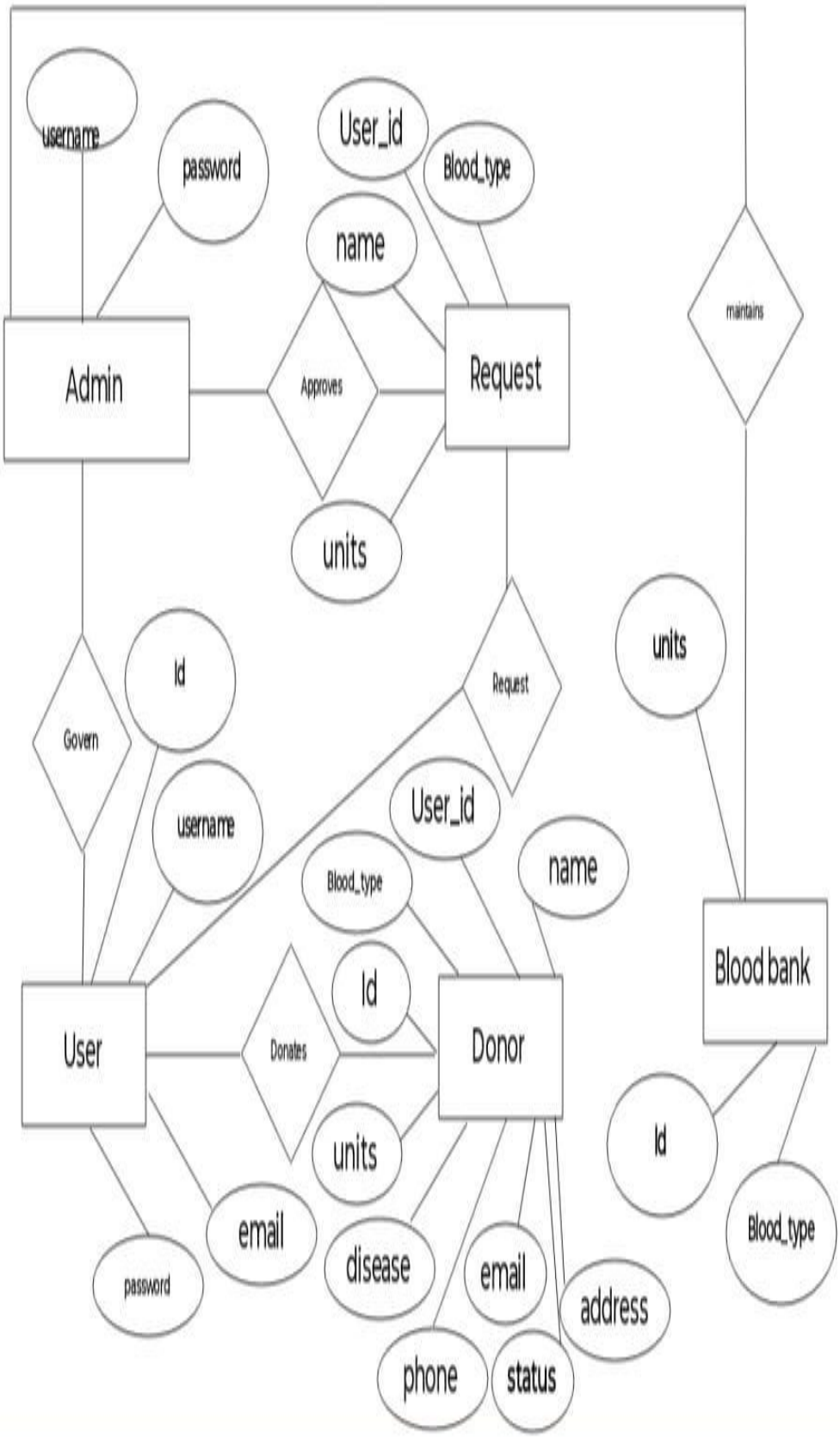
Notation	Description
	The rectangle represents an Entity.
	The diamond shape represents the relationship between the two entities.
	The oval shape represents the attributes
	The oval shape with underline represents the key attributes

Relationships:

- **Admin approves** → Request
- **Admin governs** → User
- **User requests** → Request
- **User donates** → Donor
- **Donor donates** → blood to the Blood Bank
- **Blood Bank maintains** → donated blood with units and blood type

Workflow Overview:

1. **User** registers and can either:
    - Become a **Donor** and donate blood (providing details like blood type, units, disease, etc.)
    - Make a **Request** for blood (specifying blood type and units)
2. **Admin**:
    - Governs users
    - Approves or rejects **Requests**
3. **Blood Bank**:
    - Maintains blood inventory by tracking units and blood type from donations ○ Full fills approved **Requests**


username
password
User_id
Blood_type
name
Admin
Approves
Request
maintains
units
Govern
Id
Request
units
username
User_id
Blood_type
name
Blood bank
User
Donates
Id
Donor
units
Id
email
disease
email
address
Blood_type
password
phone
status

PROGRAM CODE LISTING

USER INTERFACE

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Blood Bank Management System</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link
href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap"
rel="stylesheet">
  <style>
* {
      box-sizing: border-box;
      margin: 0;
      padding: 0;
    }

    body {
      font-family: 'Poppins', sans-serif;
      color: #fff;
overflow-x: hidden;
    }

    .slideshow {
position: fixed;       top:
0; left: 0;       width:
100%;       height:
100%;       z-index: -2;
      overflow: hidden;
    }

    .slide {
position: absolute;
width: 100%;
height: 100%;
object-fit: cover;
opacity: 0;
      transition: opacity 1s ease-in-out;
    }

    .slide.active {
opacity: 1;
```

```
  }

  header {
    position: relative;      z-index: 2;
background-color: rgba(0, 0, 0, 0.7);
padding: 16px 24px;      display: flex;
justify-content: space-between;
    align-items: center;
  }

  .logo {      font-
size: 22px;      font-
weight: 700;
    color: #ffffff;
    text-shadow: 2px 2px 4px rgba(0,0,0,0.9);
  }

  nav {
display: flex;
    gap: 16px;
  }

  nav a {      color:
white;      text-
decoration: none;
font-weight: 600;
transition: color 0.2s;
    text-shadow: 2px 2px 5px black;
  }

  nav a:hover {
color: #ffcccc;
  }

  .menu-toggle {
display: none;      flex-
direction: column;
cursor: pointer;
  }

  .menu-toggle span {
height: 3px;      width:
25px;      background:
white;      margin: 4px
0;      transition: all
0.3s;
  }
```

```
   .hero {       display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
height: 90vh;       text-
align: center;       padding:
0 20px;
     background-color: rgba(0, 0, 0, 0.4);
   }

   .hero h1 {       font-
size: 2.8rem;       margin-
bottom: 16px;       color:
#ffffff;
     text-shadow: 3px 3px 6px rgba(0,0,0,0.8), -2px -2px 4px rgba(255,255,255,0.2);
   }

   .hero p {       font-size:
1.2rem;       margin-
bottom: 28px;       color:
#ffffff;
     text-shadow: 2px 2px 5px rgba(0,0,0,0.9);
   }

   .buttons {
display: flex;
gap: 16px;
     flex-wrap: wrap;
   }

   .btn {
     padding: 12px 28px;
font-size: 16px;       border:
2px solid #fff;       border-
radius: 6px;       cursor:
pointer;
     background: rgba(255, 77, 77, 0.9);
     color: white;       font-
weight: 600;       text-
decoration: none;
     transition: all 0.3s;
     box-shadow: 2px 2px 8px rgba(0,0,0,0.5);
   }

   .btn:hover {
background: #cc0000;
     transform: scale(1.05);
   }
```

```
   @media (max-width: 768px) {
     nav {
display: none;
flex-direction:
column;
background-color:
rgba(0, 0, 0, 0.9);
       position: absolute;
top: 60px;        right:
0;        width: 200px;
       padding: 20px;
     }

     nav.active {
display: flex;
     }

     .menu-toggle {
display: flex;
     }
   }
  </style>
</head>
<body>

  <!-- Slideshow Background -->
  <div class="slideshow">
    <img src="img1.jpg" class="slide active" />
    <img src="img2.jpg" class="slide" />
    <img src="img3.jpg" class="slide" />
    <img src="img4.jpg" class="slide" />
    <img src="img5.jpg" class="slide" />
    <img src="img6.jpg" class="slide" />
    <img src="img7.jpg" class="slide" />
  </div>

  <!-- Navbar -->
  <header>
    <div class="logo">Blood Bank</div>
    <div class="menu-toggle" onclick="toggleMenu()">
      <span></span>
      <span></span>
      <span></span>
    </div>
    <nav id="navbar">
      <a href="admin_login.html">Admin Login</a>
      <a href="about.html">About</a>
```

```
      <a href="contact.html">Contact</a>
      <a href="feedback.html">Feedback</a>
      <a href="blood.html">Online</a>
    </nav>
  </header>

  <!-- Hero Section -->
  <section class="hero">
    <h1>Welcome to Blood Bank Management System</h1>
    <p>Donate Blood. Save Lives.</p>
    <div class="buttons">
      <a href="login.html" class="btn">Login</a>
      <a href="register.html" class="btn">Register</a>
    </div>
  </section>

  <script>
    const slides = document.querySelectorAll(".slide");
    let currentSlide = 0;

    function showSlide(index) {
slides.forEach((slide, i) => {
        slide.classList.toggle("active", i === index);
      });
    }

    setInterval(() => {
      currentSlide = (currentSlide + 1) % slides.length;
showSlide(currentSlide);
    }, 5000);

    function toggleMenu() {
      const navbar = document.getElementById("navbar");
      navbar.classList.toggle("active");
    }
  </script>

</body>
</html>
```

Login.php

```
<?php
$conn = new mysqli("localhost", "root", "", "blood_bank");
if ($conn->connect_error) die("Connection failed: " . $conn->connect_error);

$email = $_POST['email'];
```

```
$password = $_POST['password'];
$result = $conn->query("SELECT * FROM users WHERE email='$email'");

if ($result->num_rows == 1) {
$row = $result->fetch_assoc();
   if (password_verify($password, $row['password'])) {
session_start();
      $_SESSION['user_id'] = $row['id'];
      header("Location: dashboard.php");
      exit();
   } else {
      // Wrong password
      showError("Incorrect password. Please try again.");
   }
} else {
   // Email not registered
   showError("Email not registered.");
}

$conn->close();

function showError($message) {
echo "    <html>
   <head>
      <title>Login Error</title>
      <style>
body {
margin: 0;
padding: 0;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: linear-gradient(135deg, #f85032, #e73827);
display: flex;              justify-content: center;              align-
items: center;
            height: 100vh;
         }
         .alert-box {
background: white;
color: #e73827;
padding: 30px 40px;
border-radius: 15px;
            box-shadow: 0 8px 20px rgba(0, 0, 0, 0.2);
            text-align: center;
max-width: 400px;
animation: slideIn 0.5s ease;
         }
         h2 {
            margin-bottom: 20px;
         }
```

```
        .btn {
            background: linear-gradient(135deg, #e73827, #f85032);
color: white;              padding: 10px 25px;              margin:
10px 5px 0;              border: none;              border-radius:
30px;              cursor: pointer;              font-size: 16px;
            transition: transform 0.2s ease, background 0.3s ease;
        }
        .btn:hover {
            transform: scale(1.05);
            background: linear-gradient(135deg, #d22f1b, #fa6149);
        }
        @keyframes slideIn {
from {
                transform: translateY(-50px);
                opacity: 0;
            }
            to {
                transform: translateY(0);
                opacity: 1;
            }
        }
    </style>
  </head>
  <body>
    <div class='alert-box'>
        <h2>$message</h2>
        <button class='btn' onclick=\"location.href='register.html'\">Go to Register</button>
<button class='btn' onclick=\"location.href='login.html'\">Back to Login</button>
</div>
  </body>
  </html>
  ";
exit();
}
?>
```

Dashboard.php

```
<?php
session_start();
if (!isset($_SESSION['user_id'])) {
  header("Location: login.html");
exit;
}

$conn = new mysqli("localhost", "root", "", "blood_bank");
if ($conn->connect_error) die("Connection failed: " . $conn->connect_error);
```

```
$user_id = $_SESSION['user_id'];

// Get username from database
$user_result = $conn->query("SELECT name FROM users WHERE id = $user_id");
$user_row = $user_result->fetch_assoc();
$username = $user_row['name'];

// Fetch donation requests by user
$donation_requests = $conn->query("SELECT * FROM pending_donations WHERE user_id
= $user_id ORDER BY requested_at DESC");

// Fetch blood requests by user
$blood_requests = $conn->query("SELECT * FROM requests WHERE user_id = $user_id
ORDER BY id DESC");
?>

<!DOCTYPE html>
<html>
<head>
   <title>User Dashboard</title>
   <style>
body {
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
background-color: #fff0f0;
        margin: 0;
        padding: 30px;
      }

      .dashboard-container {
max-width: 1000px;
margin: auto;          background:
#ffffff;          padding: 40px;
border-radius: 10px;
        box-shadow: 0 0 15px rgba(204, 0, 0, 0.2);
      }
       h2 {
color: #cc0000;
        margin-bottom: 20px;
      }

      .nav-buttons {
        margin-bottom: 30px;
      }

      .nav-buttons a {
display: inline-block;
```

```
margin-right: 10px;
background-color: #cc0000;
        color: white;
padding: 10px 16px;
text-decoration: none;
border-radius: 5px;
        font-weight: bold;
    }

    .nav-buttons a:hover {
        background-color: #a30000;
    }
     table {          width:
100%;          border-collapse:
collapse;          margin-top:
20px;
        margin-bottom: 40px;
    }

    table th, table td {
padding: 12px;          border:
1px solid #ddd;          text-
align: center;
    }
     table th
{
        background-color: #f9caca;
color: #660000;
    }

    .status-button {
padding: 6px 12px;
background-color: #007bff;
color: white;          text-
decoration: none;          border-
radius: 4px;
        font-size: 13px;
    }

    .status-button:hover {
        background-color: #0056b3;
    }
  </style> </head>
<body>
<div class="dashboard-container">
  <h2>Welcome, <?= htmlspecialchars($username) ?>!</h2>

  <div class="nav-buttons">
```

```
        <a href="donate.html">Donate Blood</a>
        <a href="request_blood.html">Request Blood</a>
        <a href="logout.php">Logout</a>
    </div>

    <h3>Your Donation Requests</h3>
    <table>
        <tr>
            <th>Blood Type</th>
            <th>Units</th>
            <th>Status</th>
            <th>Requested On</th>
            <th>Action</th>
        </tr>
        <?php while ($row = $donation_requests->fetch_assoc()): ?>
        <tr>
            <td><?= htmlspecialchars($row['blood_type']) ?></td>
            <td><?= htmlspecialchars($row['units']) ?></td>
            <td><?= ucfirst(htmlspecialchars($row['status'])) ?></td>
            <td><?= htmlspecialchars($row['requested_at']) ?></td>
            <td>
                <?php if ($row['status'] === 'approved'): ?>
                    <a href="generate_certificate.php?id=<?= $row['id'] ?>" class="status-button"
target="_blank">Avail Donation Certificate</a>
                <?php else: ?>
                    <a href="user_dashboard.php" class="status-button">Check Status</a>
<?php endif; ?>
            </td>
        </tr>
        <?php endwhile; ?>
    </table>

    <h3>Your Blood Requests</h3>
    <table>
        <tr>
            <th>Blood Type</th>
            <th>Units</th>
            <th>Status</th>
            <th>Requested On</th>
            <th>Action</th>
        </tr>
        <?php while ($row = $blood_requests->fetch_assoc()): ?>
        <tr>
            <td><?= htmlspecialchars($row['blood_type']) ?></td>
<td><?= htmlspecialchars($row['units']) ?></td>
            <td><?= ucfirst(htmlspecialchars($row['status'])) ?></td>
            <td><?= htmlspecialchars($row['requested_at'] ?? '') ?></td>
```

```
        <td><a href="user_dashboard.php" class="status-button">Check Status</a></td>
      </tr>
      <?php endwhile; ?>
    </table>

</div>
</body>
</html>

<?php $conn->close(); ?>
```

Donate.php

```
<?php
session_start();
$conn = new mysqli("localhost", "root", "", "blood_bank");
if ($conn->connect_error) die("Connection failed: " . $conn->connect_error);

$user_id = $_SESSION['user_id'];
$name = $conn->real_escape_string($_POST['name']);
$age = (int)$_POST['age'];
$blood_type = $conn->real_escape_string($_POST['blood_type']);
$units = (int)$_POST['units'];
$disease = $conn->real_escape_string($_POST['disease']);
$phone = $conn->real_escape_string($_POST['phone']);
$email = $conn->real_escape_string($_POST['email']);
$address = $conn->real_escape_string($_POST['address']);

// Add status = 'pending' and timestamp
$sql = "INSERT INTO pending_donations (user_id, name, age, blood_type, units, disease, phone,
email, address, status, requested_at)
      VALUES ($user_id, '$name', $age, '$blood_type', $units, '$disease', '$phone', '$email',
'$address', 'pending', NOW())";

?>
<!DOCTYPE html>
<html>
<head>
   <title>Donation Submitted</title>
   <style>
body {
        font-family: 'Segoe UI', sans-serif;
background-color: #fff0f0;
display: flex;          justify-content:
center;          align-items: center;
        height: 100vh;
```

```
        }

        .message-box {
background: #ffffff;
padding: 40px;              border-
radius: 10px;
            box-shadow: 0 0 15px rgba(204, 0, 0, 0.3);
text-align: center;
        }
         h2
{
            color: #cc0000;
        }
         p
{
            margin: 20px 0;
            font-size: 16px;
        }

        .button {              display:
inline-block;              padding: 10px
16px;              background-color:
#cc0000;
            color: white;              text-
decoration: none;              border-
radius: 5px;
            font-weight: bold;
        }

        .button:hover {
            background-color: #a30000;
        }
    </style>
</head>
<body>
    <div class="message-box">
        <?php if ($conn->query($sql)): ?>
            <h2>Donation Submitted!</h2>
            <p>Your donation request is under review. You can check the status on your
dashboard.</p>
        <?php else: ?>
            <h2>Error</h2>
            <p><?= htmlspecialchars($conn->error) ?></p>
        <?php endif; ?>
        <a class="button" href="dashboard.php">Back to Dashboard</a>
</div>
</body>
</html>
```

```
<?php $conn->close(); ?>
```

Request.php

```
<?php
session_start();
$conn = new mysqli("localhost", "root", "", "blood_bank");
if ($conn->connect_error) die("Connection failed: " . $conn->connect_error);

$user_id = $_SESSION['user_id'];
$blood_type = $_POST['blood_type'];
$units = $_POST['units'];

$stmt = $conn->prepare("INSERT INTO requests (user_id, blood_type, units, status) VALUES (?,
?, ?, 'pending')");
$stmt->bind_param("isi", $user_id, $blood_type, $units);

?>
<!DOCTYPE html>
<html>
<head>
  <title>Request Blood</title>
  <style>
body {
        font-family: 'Segoe UI', sans-serif;
background: #fdf0f0;          display:
flex;          justify-content: center;
align-items: center;
        height: 100vh;
    }
    .message-box {
background: #fff;
padding: 30px;          border-
radius: 10px;
        box-shadow: 0 0 10px rgba(204, 0, 0, 0.2);
text-align: center;
    }
h2 {
        color: #cc0000;
    }
    a.button {
display: inline-block;
margin-top: 20px;
padding: 10px 20px;
background: #cc0000;
color: white;          text-
decoration: none;
border-radius: 6px;
```

```
            font-weight: bold;
        }
        a.button:hover {
background: #a30000;
        }
    </style>
</head>
<body>

<div class="message-box">
    <?php
    if ($stmt->execute()) {
        echo "<h2>Blood request submitted successfully!</h2>";
} else {
        echo "<h2>Error: " . htmlspecialchars($stmt->error) . "</h2>";
    }
    $stmt->close();
    $conn->close();
    ?>
    <a class="button" href="dashboard.php">← Back to Dashboard</a> </div>

</body>1
</html>
```

Admin interface

Admin_login.php

```
<?php
session_start();
$admin_user = 'admin';
$admin_pass = 'admin123'; // You can change this

if ($_POST['username'] === $admin_user && $_POST['password'] === $admin_pass) {
    $_SESSION['admin'] = true;
header("Location: admin_dashboard.php");
exit(); } else {
    showAdminError("Invalid admin login credentials.");
}

function showAdminError($message) {
    echo "
<html>
    <head>
        <title>Admin Login Error</title>
        <style>
body {
```

```
margin: 0;
padding: 0;
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
background: linear-gradient(135deg, #1f1c2c, #928dab);
display: flex;              justify-content: center;              align-items:
center;              height: 100vh;              color: #333;
        }
        .alert-box {
background: #fff;
padding: 30px 40px;
border-radius: 15px;
text-align: center;
box-shadow: 0 10px 25px
rgba(0, 0, 0, 0.2);
animation: fadeIn 0.6s ease;
        }
h2 {
            color: #d32f2f;
            margin-bottom: 20px;
        }
        .btn {
            background: linear-gradient(to right, #ff416c, #ff4b2b);
color: white;              padding: 10px 22px;              margin:
10px 8px;              border: none;              border-radius: 25px;
font-size: 15px;              cursor: pointer;
            transition: transform 0.2s ease;
        }
        .btn:hover {
            transform: scale(1.05);
        }
        @keyframes fadeIn {
            from { opacity: 0; transform: translateY(-20px); }
to { opacity: 1; transform: translateY(0); }
        }
    </style>
  </head>
  <body>
    <div class='alert-box'>
        <h2>$message</h2>
        <button class='btn' onclick=\"location.href='admin_login.html'\">Try Again</button>
<button class='btn' onclick=\"location.href='index.html'\">Back to Home</button>        </div>
  </body>
  </html>
  ";
exit();
}
?>
```

Admin_dashboard.php

```
<?php
$conn = new mysqli("localhost", "root", "", "blood_bank");
if ($conn->connect_error) die("Connection failed: " . $conn->connect_error);

$totalUsers = $conn->query("SELECT COUNT(*) AS total FROM users")->fetch_assoc()['total'];
$bloodStock = $conn->query("SELECT * FROM blood_stock");
$pendingDonations = $conn->query("SELECT * FROM pending_donations WHERE status = 'pending'");
$pendingRequests = $conn->query("SELECT * FROM requests WHERE status = 'pending'");
$donors = $conn->query("SELECT * FROM donations");
?>

<!DOCTYPE html>
<html>
<head>
  <title>Admin Dashboard</title>
  <style>
body {
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
background-color: #fdf0f0;
        margin: 0;
padding: 30px;
color: #333;
      }

      h2, h3 {
        color: #b30000;
      }

      .dashboard-header {
text-align: center;        margin-bottom: 40px;        position: relative;
      }

      .action-buttons {
position: absolute;
        top: 0;
        right: 0;
        margin: 10px 20px;
      }
```

```
        .action-buttons a {
text-decoration: none;
padding: 8px 16px;            margin-
left: 10px;            border-radius:
6px;            font-weight: bold;
font-size: 14px;
background-color: #b71c1c;
            color: white;
            transition: background-color 0.3s ease;
        }

        .action-buttons a:hover {
```

```
    }

        background-color: #8b0000;


    .card-container {
display: flex;          flex-
wrap: wrap;          gap:
20px;
        margin-bottom: 40px;
    }

    .box {
        flex: 1;          min-width: 200px;
background: #fff;          border: 2px solid
#f2dada;          padding: 20px;          border-
radius: 12px;          box-shadow: 0 4px 8px
rgba(204, 0, 0, 0.1);
        text-align: center;
    }

    .box h3 {
margin: 10px 0;
        color: #cc0000;
    }

    .box p {
font-size: 24px;
color: #333;
        font-weight: bold;
    }
     table {          width:
100%;          border-collapse:
collapse;
        margin-bottom: 40px;
    }
    th {
        background-color: #ffcccc;
        color: #800000;
    }       th, td {
padding: 12px;          border:
1px solid #ddd;          text-
align: center;
    }

    tr:nth-child(even) {
        background-color: #fff5f5;
```

```
        }

        a {             text-
decoration: none;
padding: 6px 12px;
margin: 0 2px;
border-radius: 5px;
font-weight: bold;
            font-size: 13px;
        }

        a:hover {
opacity: 0.85;
        }

        a[href*="approve"] {
background-color: #28a745;
color: white;
        }

        a[href*="deny"] {
background-color: #dc3545;
color: white;
        }

        a[href*="delete"] {
background-color: #6c757d;
color: white;
        }

        a[href*="edit"] {
background-color: #007bff;
color: white;
        }

        @media (max-width: 768px) {             .card-
container {
                flex-direction: column;
            }

            .action-buttons {
position: static;
margin-top: 10px;               text-
align: center;
            }

            .action-buttons a {
display: inline-block;
margin: 5px;
```

```
        }

            }

    </style>
</head>
<body>

<div class="dashboard-header">
    <h2>  Admin Dashboard - Blood Bank Management</h2>
    <div class="action-buttons">
        <a href="admin_login.html">  Back</a>
        <a href="logout.php">  Logout</a>
    </div>
</div>

<div class="card-container">
    <div class="box">
        <h3>Total Users</h3>
        <p><?= $totalUsers ?></p>
    </div>
```

Admin_approve_request.php

```
<?php
session_start(); if
(!isset($_SESSION['admin'])) {
die("Access denied");
}

$conn = new mysqli("localhost", "root", "", "blood_bank");
if ($conn->connect_error) die("Connection failed: " . $conn->connect_error);

$id = intval($_GET['id']);
$action = $_GET['action'] ?? '';

if (!$id || !in_array($action, ['approve', 'deny', 'delete'])) {
die("Invalid request");
}

// Fetch the request details
$stmt = $conn->prepare("SELECT * FROM requests WHERE id = ?");
$stmt->bind_param("i", $id);
$stmt->execute();
$request = $stmt->get_result()->fetch_assoc();
$stmt->close();

if (!$request) {
```

```
        }

    die("Request not found");
}
```

```
$blood_type = $request['blood_type'];
$units = intval($request['units']);

function showMessage($title, $message, $icon = ' ') {
   echo <<<HTML
<!DOCTYPE html>
<html>
<head>
   <title>{$title}</title>     <style>
body {             background-color: #fff5f5;
font-family: 'Segoe UI', sans-serif;
          display: flex;
justify-content: center;
align-items: center;
height: 100vh;
margin: 0;
       }
       .message-box {
background-color: #ffffff;
border: 2px solid #ffcccc;
padding: 40px;             border-
radius: 12px;
          box-shadow: 0 0 15px rgba(204, 0, 0, 0.2);
          text-align: center;
          max-width: 400px;
       }
       .message-box h2 {
color: #cc0000;
          margin-bottom: 20px;
       }
       .message-box p {
font-size: 18px;
color: #333;
          margin-bottom: 30px;
       }
       .btn {             padding: 12px
20px;             background-color:
#cc0000;             color: white;
text-decoration: none;
border-radius: 6px;             font-
weight: bold;
          transition: background-color 0.3s ease;
       }
       .btn:hover {
          background-color: #a30000;
       }
   </style>
</head>
```

```
<body>
    <div class="message-box">
        <h2>{$icon} {$title}</h2>
        <p>{$message}</p>
        <a href="admin_dashboard.php" class="btn">Back to Dashboard</a>
</div>
</body>
</html>
HTML;
    exit;
}

if ($action === 'approve') {
    $stmt = $conn->prepare("SELECT units FROM blood_stock WHERE blood_type = ?");
$stmt->bind_param("s", $blood_type);
    $stmt->execute();
    $stock = $stmt->get_result()->fetch_assoc();
    $stmt->close();

    if (!$stock || $stock['units'] < $units) {
        showMessage("Insufficient Stock", "Not enough blood units available to approve the
request.", " ");
    }

    $stmt = $conn->prepare("UPDATE blood_stock SET units = units - ? WHERE blood_type
= ?");
    $stmt->bind_param("is", $units, $blood_type);
    $stmt->execute();
    $stmt->close();

    $stmt = $conn->prepare("UPDATE requests SET status = 'approved' WHERE id = ?");
$stmt->bind_param("i", $id);
    $stmt->execute();
    $stmt->close();

    showMessage("Request Approved", "The request for <strong>$units</strong> unit(s) of
<strong>$blood_type</strong> blood has been approved."); }

if ($action === 'deny') {
    $stmt = $conn->prepare("UPDATE requests SET status = 'denied' WHERE id = ?");
    $stmt->bind_param("i", $id);
    $stmt->execute();
    $stmt->close();

    showMessage("Request Denied", "The blood request has been denied.", " "); }

if ($action === 'delete') {
    $stmt = $conn->prepare("DELETE FROM requests WHERE id = ?");
```

```
    $stmt->bind_param("i", $id);
    $stmt->execute();
    $stmt->close();

    showMessage("Request Deleted", "The blood request has been successfully deleted from
the system.", " ");
}

$conn->close();
header("Location: admin_dashboard.php"); exit;
```

admin_approve_donation.php

```
<?php
$conn = new mysqli("localhost", "root", "", "blood_bank");
if ($conn->connect_error) die("Connection failed: " . $conn->connect_error);

$id = intval($_GET['id']); // donation request ID
$action = $_GET['action']; // approve or deny

function showMessage($title, $message, $icon = ' ') {     echo
<<<HTML
<!DOCTYPE html>
<html>
<head>
    <title>{$title}</title>
    <style>
body {
            background-color: #fff5f5;
font-family: 'Segoe UI', sans-serif;
            display: flex;
justify-content: center;
align-items: center;
height: 100vh;
margin: 0;
        }
        .message-box {
            background-color: #ffffff;
border: 2px solid #ffcccc;
padding: 40px;              border-
radius: 12px;
            box-shadow: 0 0 15px rgba(204, 0, 0, 0.2);
text-align: center;
            max-width: 400px;
        }
        .message-box h2 {
color: #cc0000;
margin-bottom: 20px;         }
```

```
        .message-box p {
font-size: 18px;
color: #333;
            margin-bottom: 30px;
        }
        .btn {
            padding: 12px 20px;
background-color: #cc0000;
            color: white;            text-
decoration: none;            border-
radius: 6px;            font-weight:
bold;
            transition: background-color 0.3s ease;
        }
        .btn:hover {
            background-color: #a30000;
        }
    </style>
</head>
<body>
    <div class="message-box">
        <h2>{$icon} {$title}</h2>
        <p>{$message}</p>
        <a href="admin_dashboard.php" class="btn">Back to Dashboard</a>
</div>
</body>
</html>
HTML;
    exit;
}

// Get pending donation entry
$donation = $conn->query("SELECT * FROM pending_donations WHERE id = $id")-
>fetch_assoc();

if (!$donation) {
    showMessage("Donation Not Found", "The donation entry with ID $id does not exist.",
" ");
}

if ($action === 'approve') {
// Insert into donations table
    $stmt = $conn->prepare("INSERT INTO donations (name, age, blood_type, units, disease,
phone, email, address) VALUES (?, ?, ?, ?, ?, ?, ?, ?)");
    $stmt->bind_param(
        "sisisssss",
        $donation['name'], $donation['age'], $donation['blood_type'], $donation['units'],
        $donation['disease'], $donation['phone'], $donation['email'], $donation['address']
```

```
    );
    $stmt->execute();
    $stmt->close();

    // Update blood_stock
    $blood_type = $donation['blood_type'];
    $units = $donation['units'];
    $check = $conn->query("SELECT * FROM blood_stock WHERE blood_type =
'$blood_type'");
    if ($check->num_rows > 0) {
        $conn->query("UPDATE blood_stock SET units = units + $units WHERE blood_type =
'$blood_type'");
    } else {
        $conn->query("INSERT INTO blood_stock (blood_type, units) VALUES
('$blood_type', $units)");
    }

    // Update pending_donations status
    $conn->query("UPDATE pending_donations SET status = 'approved' WHERE id = $id");

    showMessage("Donation Approved", "Donation of <strong>$units</strong> unit(s) of
<strong>$blood_type</strong> blood has been approved and added to inventory.");
} elseif ($action === 'deny') {
    $conn->query("UPDATE pending_donations SET status = 'denied' WHERE id = $id");
showMessage("Donation Denied", "The donation request has been denied.", " ");
} else {
    showMessage("Invalid Action", "The specified action is not valid.", " "); }

$conn->close();
?>
```

Edit_donor.php

```
<?php
$conn = new mysqli("localhost", "root", "", "blood_bank"); if ($conn-
>connect_error) die("Connection failed: " . $conn->connect_error);

$id = $_GET['id'];

$result = $conn->query("SELECT * FROM donations WHERE id = $id");
if ($result->num_rows == 0) {     die("Donor not found.");
}
$donor = $result->fetch_assoc();

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $new_name = $_POST['name'];
    $new_age = $_POST['age'];
```

```
    $new_blood_type = $_POST['blood_type'];
    $new_units = $_POST['units'];
    $new_disease = $_POST['disease'];
    $new_phone = $_POST['phone'];
    $new_email = $_POST['email'];
    $new_address = $_POST['address'];

    $old_type = $donor['blood_type'];
    $old_units = $donor['units'];

    if ($old_type !== $new_blood_type || $old_units != $new_units) {
        $conn->query("UPDATE blood_stock SET units = units - $old_units WHERE
blood_type = '$old_type'");
        $exists = $conn->query("SELECT * FROM blood_stock WHERE blood_type =
'$new_blood_type'");
        if ($exists->num_rows > 0) {
            $conn->query("UPDATE blood_stock SET units = units + $new_units WHERE
blood_type = '$new_blood_type'");
        } else {
            $conn->query("INSERT INTO blood_stock (blood_type, units) VALUES
('$new_blood_type', $new_units)");
        }
    }
    $update = $conn->prepare("UPDATE donations SET name=?, age=?, blood_type=?,
units=?, disease=?, phone=?, email=?, address=? WHERE id=?");
    $update->bind_param("sisissssi", $new_name, $new_age, $new_blood_type, $new_units,
$new_disease, $new_phone, $new_email, $new_address, $id);
    $update->execute();
    // Show success message and styled return
link     echo "     <div style='         max-width:
500px;         margin: 60px auto;
background: #e8f5e9;         padding: 25px;
border-radius: 10px;         text-align: center;
box-shadow: 0 0 15px rgba(0, 128, 0, 0.1);
border: 2px solid #81c784;
        font-family: \"Segoe UI\", Tahoma, Geneva, Verdana, sans-serif;
    '>
      <h2 style='color: #2e7d32;'>  Donor updated successfully!</h2>
        <p>
            <a href='admin_dashboard.php' style='
                display: inline-block;
margin-top: 15px;                 padding:
10px 20px;                 background-
color: #388e3c;
                color:                 white;
text-decoration:                 none;
border-radius:                   6px;
```

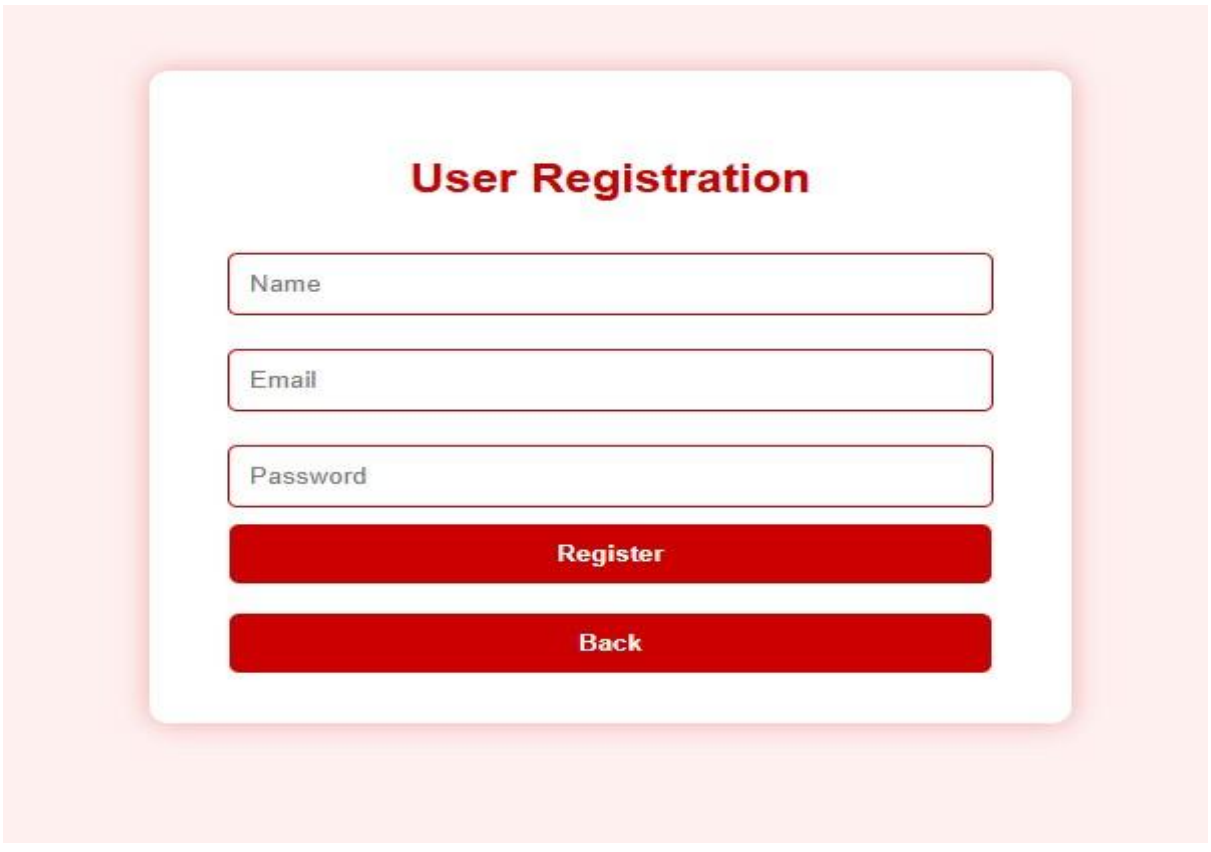
```
transition:       background-color
0.3s ease;
              font-weight: bold;
          '           onmouseover='this.style.backgroundColor=\"#2e7d32\"'
onmouseout='this.style.backgroundColor=\"#388e3c\"'>
              ← Back to Admin Dashboard
          </a>
       </p>
   </div>";
   $conn->close();
exit;
}
?>

<!DOCTYPE html>
<html>
<head>
   <title>  Edit Donor - Blood Bank System</title>
   <style>
body {
          font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
background: #fff5f5;
          color: #333;
          padding: 40px;
       }
        h2 {            text-
align: center;
          color: #c62828;
       }

       form {            max-
width: 500px;
margin: 30px auto;
background: #fff;
padding: 25px;
border-radius: 10px;
          box-shadow: 0 0 15px rgba(198, 40, 40, 0.1);
          border: 2px solid #ffcdd2;
       }

       label {
          font-weight: bold;
display: block;
          margin: 10px 0 5px;
       }
       input[type="text"],
input[type="number"],
```

```
        input[type="email"],
select,         textarea {
            width: 100%;
padding: 10px;
margin-bottom: 12px;
border: 1px solid #ccc;
border-radius: 6px;
            font-size: 15px;
        }
        input[type="submit"] {
            background-color: #d32f2f;
            color: #fff;
border: none;
padding: 12px 18px;
border-radius: 6px;
cursor: pointer;
font-size: 16px;
            transition: background-color 0.3s ease;
        }
        input[type="submit"]:hover {
            background-color: #b71c1c;
        }         .note {
text-align: center;
            margin-top: 20px;
        }
        .note a {
color: #d32f2f;
            text-decoration: none;
        }
        .note a:hover {
            text-decoration: underline;
        }
    </style>
</head>
<body>
<h2>  Edit Donor Information</h2>

<form method="POST">
    <label>Name:</label>
    <input type="text" name="name" value="<?= htmlspecialchars($donor['name']) ?>"
required>

    <label>Age:</label>
    <input type="number" name="age" value="<?= $donor['age'] ?>" required>

    <label>Blood Type:</label>
    <select name="blood_type" required>
        <?php
```

```
        $bloodTypes = ['A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-'];
foreach ($bloodTypes as $type) {
            $selected = ($type == $donor['blood_type']) ? "selected" : "";
            echo "<option value='$type' $selected>$type</option>";
        }
        ?>
    </select>

    <label>Units:</label>
    <input type="number" name="units" value="<?= $donor['units'] ?>" required>

    <label>Disease:</label>
    <input type="text" name="disease" value="<?= htmlspecialchars($donor['disease']) ?>">

    <label>Phone:</label>
    <input type="text" name="phone" value="<?= htmlspecialchars($donor['phone']) ?>">

    <label>Email:</label>
    <input type="email" name="email" value="<?= htmlspecialchars($donor['email']) ?>">

    <label>Address:</label>
    <textarea name="address" rows="3"><?= htmlspecialchars($donor['address'])
?></textarea>

    <input type="submit" value="Update Donor">
</form>

<div class="note">
    <a href="admin_dashboard.php">← Back to Dashboard</a>
</div>

</body>
</html
```

5.USER INTERFACE


Blood Bank
Admin Login
About
Contact
Feedback
Online
Welcome to Blood Bank Management System
Donate Blood. Save Lives.
Login
Register


User Registration
Name
Email
Password
Register
Back

Welcome, Yashwith K!

Donate Blood | Request Blood | Logout

Your Donation Requests

Blood Type	Units	Status	Requested On	Action
A+	30	Approved	2025-05-18 13:00:12	Avail Donation Certificate
A+	40	Approved	2025-05-18 12:44:23	Avail Donation Certificate

Your Blood Requests

Blood Type	Units	Status	Requested On	Action
A+	11	Pending	2025-05-18 20:43:18	Check Status
A+	40	Approved	2025-05-18 13:01:19	Check Status
B+	30	Denied	2025-05-18 13:01:10	Check Status
A+	10	Approved	2025-05-18 12:44:38	Check Status

Donate Blood

Full Name

Age

Select Blood Type

Units

Any Disease (if none, type 'None')

Phone

Email

Address

Donate

Back To Dashboard

Request Blood

Select Blood Type

Units Required

Submit Request

Back to Dashboard

Welcome, User!

Back **Logout**

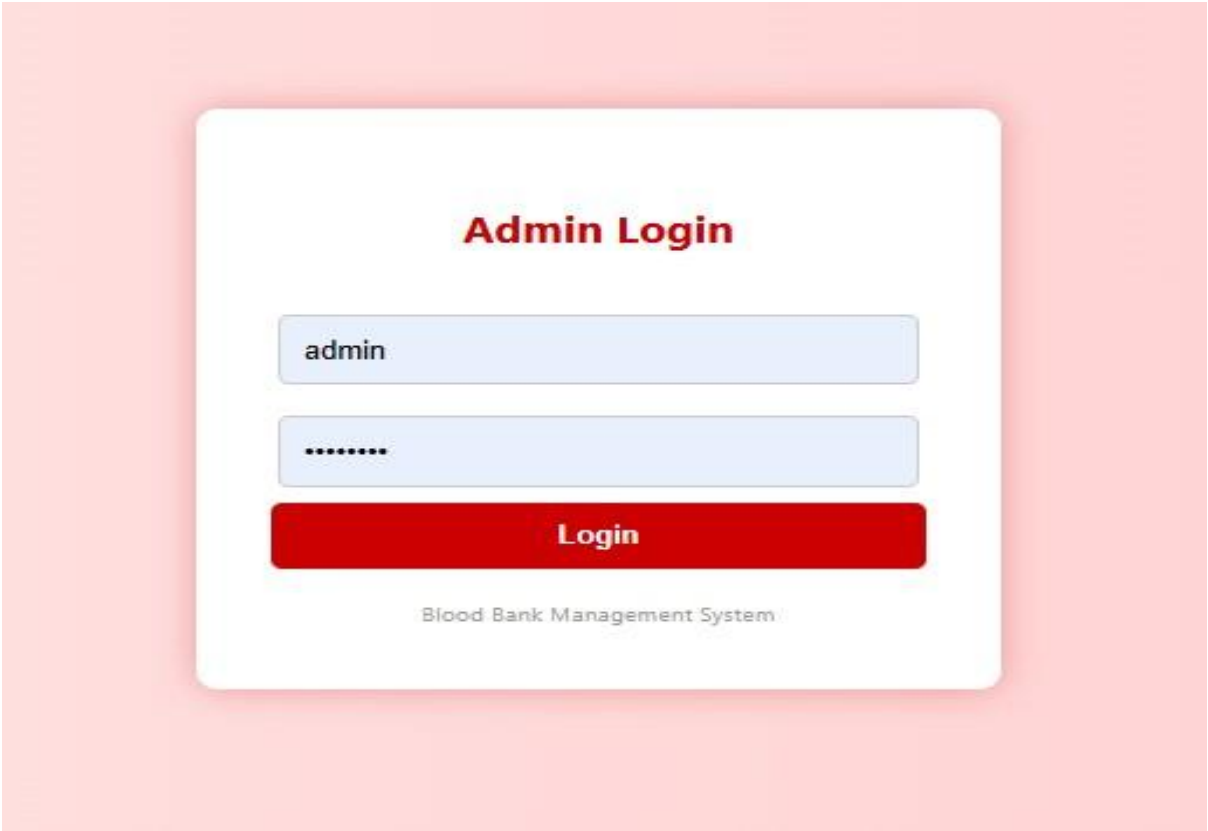
Your Donation Requests

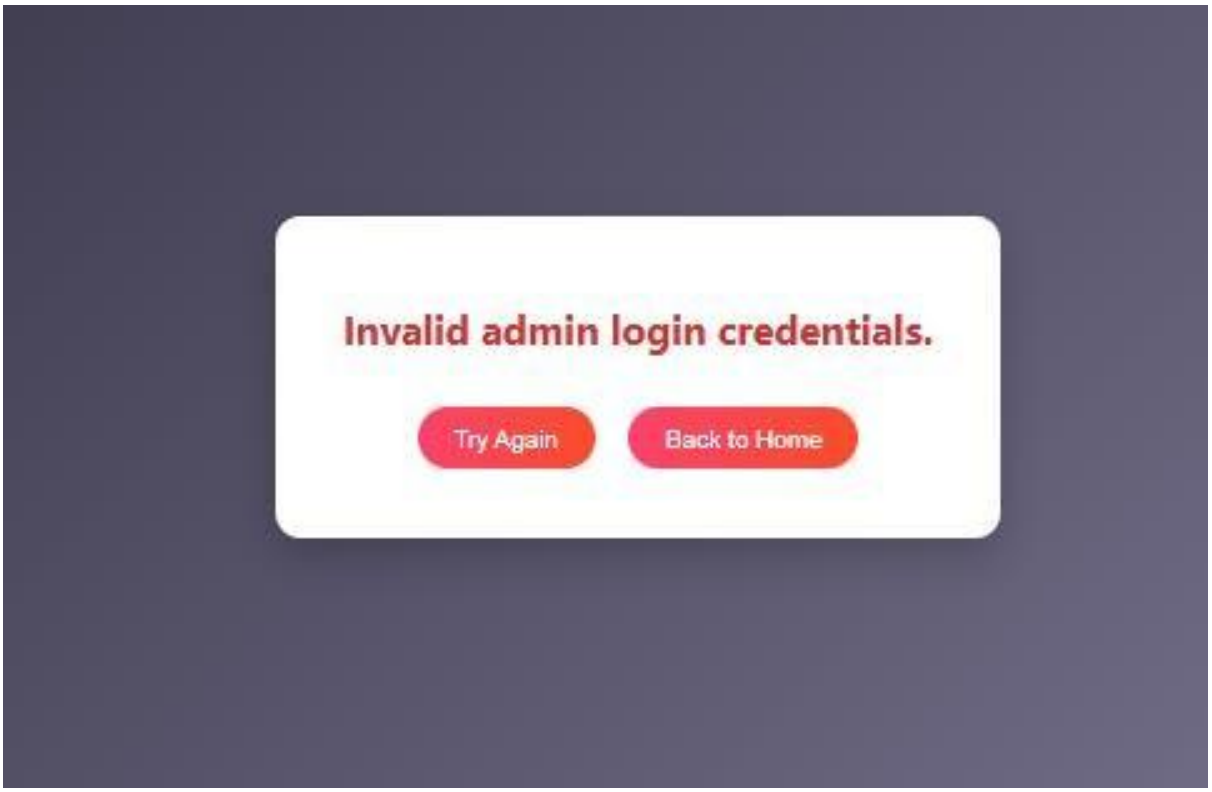
Blood Type	Units	Status	Date
A+	30	Approved	2025-05-18 13:00:12
A+	40	Approved	2025-05-18 12:44:23

Your Blood Requests

Blood Type	Units	Status	Date
A+	11	Pending	2025-05-18 20:43:18
A+	40	Approved	2025-05-18 13:01:19
B+	30	Denied	2025-05-18 13:01:10
A+	10	Approved	2025-05-18 12:44:38


CERTIFICATE OF APPRECIATION
This certificate is proudly awarded to
YASHWITH K
For your generous donation of blood and contribution to saving lives.
Your act of kindness and humanity is deeply appreciated.
You are a real-life hero! Thank you for making a difference.
Issued on: May 18, 2025
Blood Bank Administrator
Download as PDF


Admin Login
admin
Login
Blood Bank Management System



Admin Dashboard - Blood Bank Management

Back Logout

Total Users	A+
1	20 units

Pending Donation Requests

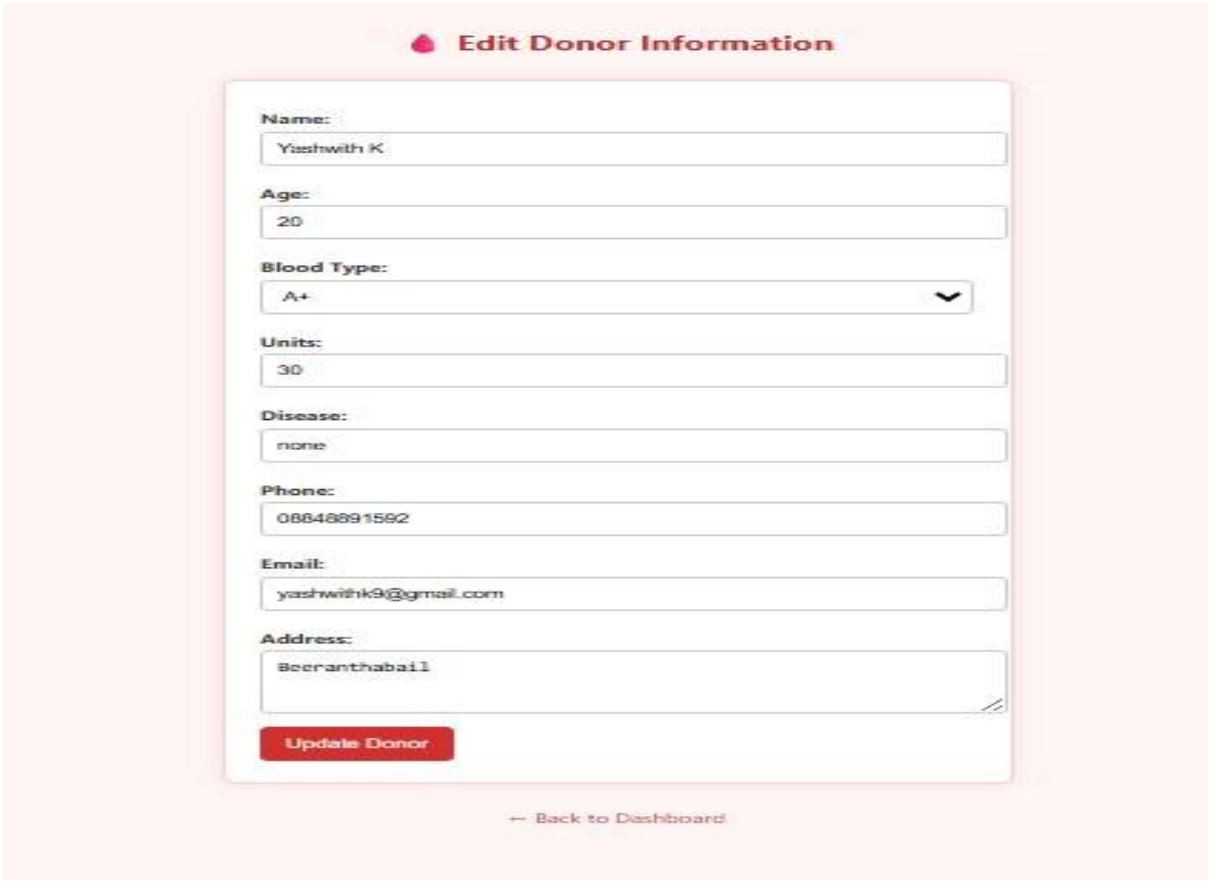
Name	Blood Type	Units	Status	Actions

Pending Blood Requests

User ID	Blood Type	Units	Status	Actions
19	A+	11	Pending	Approve Deny

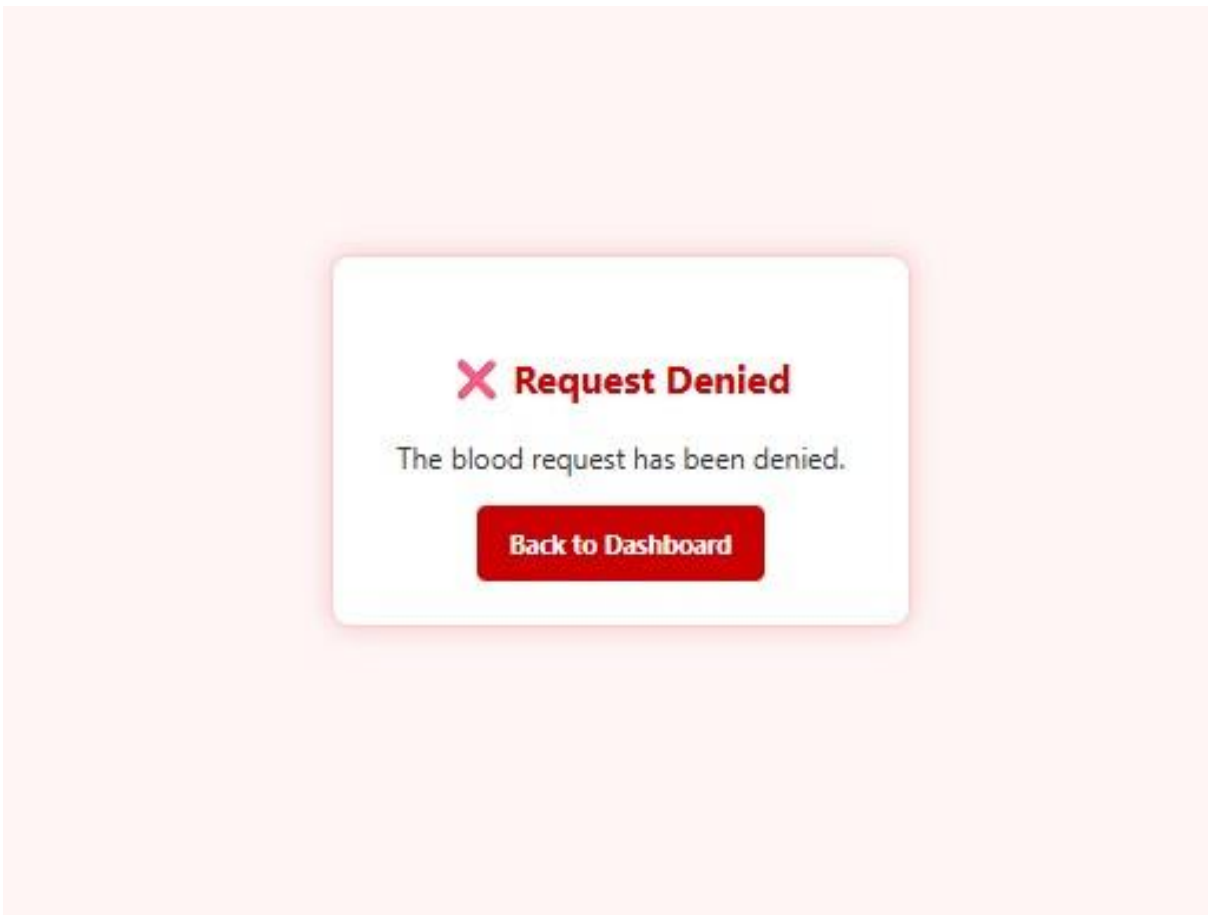
Registered Donors

Name	Blood Type	Units	Email	Action
Yashwith K	A+	30	yashwithk9@gmail.com	Edit Delete


Edit Donor Information
Name:
Yashwith K
Age:
20
Blood Type:
A+
Units:
30
Disease:
none
Phone:
08848891592
Email:
yashwithk9@gmail.com
Address:
Beeranthabail
Update Donor
← Back to Dashboard


Donor updated successfully!
← Back to Admin Dashboard


Request Approved
The request for 11 unit(s) of A+ blood has been approved.
Back to Dashboard


Request Denied
The blood request has been denied.
Back to Dashboard

Donation Submitted!

Your donation request is under review. You can check the status on your dashboard.

Back to Dashboard

⚠ Insufficient Stock

Not enough blood units available to approve the request.

Back to Dashboard

Back to Home

Contact Us

We're here to help you 24/7 — for emergencies, support, and blood availability

Emergency Services

- Ambulance: **102**
- Police: **100**
- Fire Services: **101**
- Nearest Emergency Hospital: **+91-9876676732**

YMVP BloodCare Support

- Email: YMVP@bloodcare.org
- Helpline: **1800-121-1234**
- Hours: 24/7 Emergency Support

Hospital Info

- Name: YMVP Cross Medical Hospital
- Address: 123 Main Road, kasaragod,Kerala, India

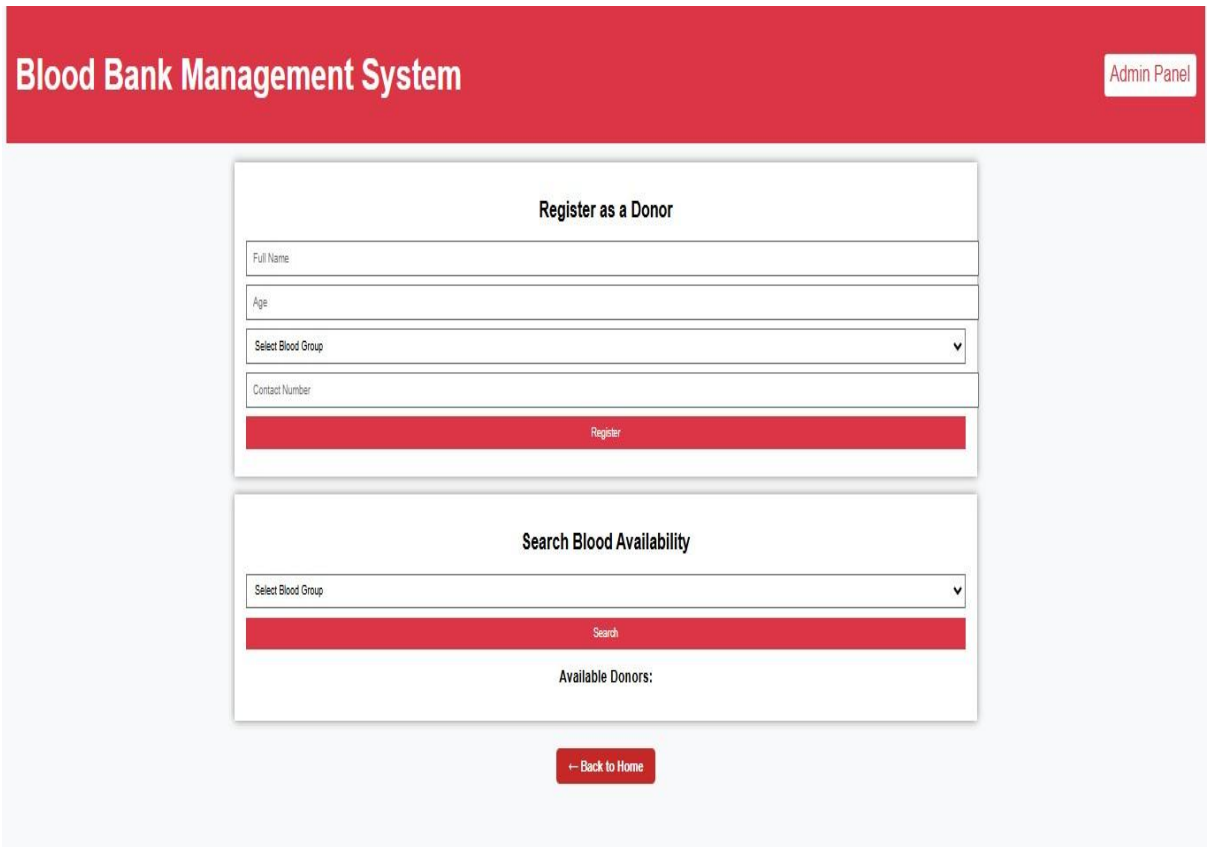
Blood Units Availavle

Technical & Website Queries

- Developer Team: YMVP Web Team
- Email: YMVP@bloodcare.org
- Contact: +91-7000-111-222

For URGENT blood requests, call our emergency helpline: 1800-121-1234


Feedback
Name
Email
Message
Submit Feedback


Blood Bank Management System
Admin Panel
Register as a Donor
Full Name
Age
Select Blood Group
Contact Number
Register
Search Blood Availability
Select Blood Group
Search
Available Donors:
← Back to Home

6.Testing

TESTING

Introduction

Testing is an essential aspect of software development that ensures the quality, reliability, and functionality of software products. It involves systematically examining and evaluating software components, features, and systems to identify defects, errors, or discrepancies between expected and actual outcomes. Through a structured and methodical approach, testing aims to validate that the software meets specified requirements, functions correctly under various conditions, and delivers a positive user experience.

Software testing is a critical component of the software development process, ensuring that software meets quality standards and performs as expected. It involves the systematic execution of programs or applications with the intent of finding errors and verifying that they function correctly. Testing encompasses various techniques, methodologies, and tools to identify defects, validate functionality, and ensure reliability, security, and performance.

The primary goals of software testing include:

1.Validating Requirements: Testing ensures that software meets the specified requirements and functions according to user expectations

.

2.Identifying Defects: Testing helps uncover defects or bugs in the software, ranging from syntax errors to logical flaws, ensuring that they are addressed before deployment.

3.Ensuring Reliability: Testing verifies the reliability and stability of software under different conditions, ensuring it performs consistently and does not fail unexpectedly.

4.Improving Quality: By identifying and fixing defects early in the development process, testing contributes to the overall quality of the software, leading to increased user satisfaction.

5.Mitigating Risks: Testing helps mitigate risks associated with software failures, security vulnerabilities, and performance bottlenecks, ensuring a smoother deployment and operation.
Software testing comprises various types, including unit testing, integration testing, system testing, acceptance testing, performance testing, security testing, and usability testing. Each type focuses on different aspects of the software, ranging from individual components to the entire system, and employs specific techniques and tools tailored to its objectives.

In summary, software testing is indispensable for delivering high-quality software that meets user requirements, functions reliably, and performs optimally. It is an iterative process integrated throughout the software development lifecycle, ensuring that defects are identified and addressed early, ultimately leading to the successful delivery of software products and services. 88

ii. Test Reports

Test reports are comprehensive documents that provide a detailed overview of the testing process, results, and findings across different stages of software development. Here's a breakdown of what each section typically includes:

Unit Testing:

1.This section focuses on testing individual components or units of the software in isolation.
2.It includes descriptions of the units tested, the test cases executed, and the outcomes of those tests.
3.Each unit test should cover specific functionalities or features of the component under test.
4.The report may also include metrics such as code coverage to assess the effectiveness of the unit tests in covering the codebase

.

Admin Login :

Test No.	Test Description	Expected Output	Result
1	If the admin name is not entered	Enter the name and the password	Success
2	If the password is not entered	Please enter the correct password	Success

3	If the name is not entered	Please enter the correct name	Success

Customer register page and login:

Test No.	Test Description	Test Output	Result
1.	Register	The entered data is stored in database and the site is redirected to login page	Success
2.	Login	The site is redirected to dashboard page page	Success

Integration Testing:

1.This part of the report discusses testing the interactions between different units/modules of the software.
2.It outlines the integration test cases executed, including the inputs provided and the expected outputs.
3.Integration testing ensures that the individual components work together as expected when integrated into the larger system.
4.Issues such as communication failures, data mismatches, or interface problems may be highlighted in this section.

Admin Index Page:

Test No.	Test Description	Expected Output	Result
1.	Donation request	The site is redirected to manage blood requests.	Success
2.	Blood Request	The site is redirected to manage requested blood.	Success
3.	Edit user	The site is redirected to edit user data and updates it	Success

4.	Delete user	The site is redirected to delete user data	Success
5.	Blood stock	Shows the blood type and unit stock	Success
7.	Login out	The site is redirected to index page	Success

Customer Index Page:

Test No.	Test Description	Expected Output	Result
1.	Donate	Customer can donate blood by providing data	Success
2.	Request	Customer can request blood by providing data	Success
3.	Check status	Customer can check blood donation and request status .	Success
4.	Avail Certificate	Customer can avail certificate after admin approve blood donation.	Success
5.	logout	The site is redirected to index page	Success

System Testing

System Testing is the testing of complete and fully integrated software products. Usually, software is only one element of larger computer-based system. Ultimately, software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

System testing tables:

Sl. No	**Test condition**	**Test report**
1.	System lading	Successful
2.	System run procedure	Successful
3.	File I/O operation	Successful
4.	Database communication	Successful
5.	Server/customer interaction	Successful
6.	Memory usage	Normal
7.	System processor usage	Normal
8.	Authentication/Authorization	Successful

7.CONCLUSION

CONCLUSION

The project work entitled "**BLOOD BANK MANAGEMENT SYSTEM"** has been designed using JAVASCRIPT, CSS, HTML as front end and PHP and MySQL, as back end. Blood bank management system is a web application designed to manage and streamline the operations of a blood bank.it helps in maintaining detailed records of blood donors, blood inventory, blood requests and transaction history and provides tools for administration.

8.Limitations

Limitations

Quality Assurance: Ensuring the quality of second-hand items can be challenging.
Limited reporting and analytics basic systems lacks advanced reporting tools or data analytics to predict shortage of donor trends .
Online Blood bank management system needs a stable internet connection, which can be a problem with poor network coverage.
4.Content overflow: if the content includes section breaks, tables or images, pasting might shift layout or push content to a different page.

9.Scope of enhancement

Scope of Enhancement

Advanced Search & Filter

Enhancement: Allow users and admins to filter donors/requests by blood type, location, last donation date, and availability.

Benefit: Speeds up matching process and improves usability.

SMS & Email Notifications

Enhancement: Notify users/admins via SMS/email for:

Request status (approved/denied)

Donation reminders

Blood stock alerts

Benefit: Keeps users informed and engaged.

Integration with Hospitals and NGOs

Enhancement: Enable API-based data sharing with hospitals, blood banks, and NGOs

.

Benefit: Creates a wider and more efficient donation network.

Mobile App Integration

Enhancement: Develop a companion mobile app for users to register, donate, and track status.

Benefit: Improves accessibility and user engagement.

Analytics and Reporting Dashboard

Enhancement: Provide graphical reports on:

Blood type demand/supply

Donation trends

Request fulfillment rates

Benefit: Helps in decision-making and strategic planning.

10.ABBREVIATIONS & ACRONYMS

ABBREVIATIONS AND ACRONYMS

SRS -Software Requirements Specification

CFD -Context Flow Diagram

DFD -Data Flow Diagram

ERD -Entity Relationship Diagram

ADMIN -The Administration

PHP -Hypertext Pre-Processor

HTML -Hypertext Markup Language

CSS -Cascading Style Sheets

SQL -Structured query language

RAM -Random Access Memory

11.BIBILOGRAPHY / REFERENCES

BIBILOGRAPHY/REFERENCES

Google

You tube

ChatGPT