CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Blood donation management encompasses the coordination, organization, and optimization of processes involved in blood collection, storage, distribution, and utilization. It plays a critical role in ensuring an adequate and safe blood supply for medical treatments, surgeries, and emergencies. An effective blood donation management system integrates various stakeholders including blood banks, hospitals, donors, and recipients to streamline operations and maximize the impact of blood donations.

1.2 PROBLEM STATEMENT

The current blood management system faces several challenges that hinder its efficiency and effectiveness in ensuring an adequate and safe blood supply for medical treatments and emergencies

1.3 DATABASE MANAGEMENT SYSTEM

A database management system (DBMS) is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data. The DBMS essentially serves as an interface between the database and end users application programs, ensuring that data is consistently organized and remains easily accessible.

The DBMS manages three important things: the data, the database engine that allows data to be accessed, locked and modified, and the database schema, which defines the database's logical structure. These three foundational elements help to provide concurrency, security, data integrity and uniform administration procedures. Typical database administration tasks supported by the DBMS include change management, performance monitoring/tuning and backup and recovery. Many database management systems are also responsible for automated rollbacks, restarts and recovery as well as the logging and auditing of activity

1.4 SQL

SQL is a standard language for storing, manipulating and retrieving data in databases. Originally based upon relational algebra and tuple relational calculus, SQL consists of data definition language, data manipulation language, and data control language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control. SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987.[13]Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

1.5 HTML/JavaScript

HTML is a markup language used for structuring and presenting content on the web and the fifth and current major version of the HTML standard. HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications. JavaScript often abbreviated as JS, is a high-level, interpreted programming language. It is a language which is also characterized as dynamic, weakly typed, proto type based and multi-paradigm. Alongside HTML and CSS, JavaScript is one of the three core technologies of the World Wide Web. JavaScript enables interactive web pages and thus is an essential part of web applications. The vast majority of websites use it, and all major web browsers have a dedicated JavaScript engine to execute it.

CHAPTER 2

REQUIREMENTS SPECIFICATION

2.1 OVERALL DESCRIPTION

The college database management system is a centralized platform designed to handle diverse aspects of college operations efficiently. It serves as a comprehensive repository for managing student data, faculty information, course details, schedules, grades, and administrative tasks. The system facilitates streamlined processes for student enrollment, course registration, academic planning, grading, and financial transactions. Through user-friendly interfaces, stakeholders such as administrators, faculty, staff, and students can access relevant information and perform necessary tasks. Security measures are implemented to safeguard sensitive data, ensuring compliance with privacy regulations. Overall, the system aims to enhance organizational effectiveness, improve data accuracy, and promote seamless communication within the college community.

2.2 SPECIFIC REQUIREMENTS

2.2.1 SOFTWARE REQUIREMENTS

- Web server (e.g., Apache)
- PHP 7.x
- MySQL 5.x
- HTML, CSS, Bootstrap for frontend
- Ajax, Query for dynamic content
- Modal for displaying messages

2.2.2 HARDWARE REQUIREMENTS

- Processor: Intel Core i5 or equivalent
- RAM: 8GB or higher
- Storage: 500GB HDD or SSD
- Network: Ethernet or Wi-Fi connection
- Display: 1080p resolution or higher
- Server to host the web application
- Client devices (e.g., computers, smartphones, tablets) to access the application

2.2.3 TECHNOLOGY

- PHP for server-side scripting
- MySQL for database management
- HTML5, CSS3, Bootstrap 4 for frontend development
- JavaScript, Ajax, jQuery for dynamic content loading
- Modal for displaying messages and forms

CHAPTER 3

DETAILED DESIGN

3.1 SYSTEM DESIGN

Designing a detailed database for blood donation management involves structuring the data to efficiently store and manage information related to donors, donations, blood inventory, recipients, and other relevant entities. Below is a proposed schema for a relational database:

Database Design Considerations:

1. Normalization:
   - Ensure the database is normalized to minimize redundancy and maintain data integrity.
   - Apply normalization techniques such as 1NF, 2NF, and 3NF to eliminate data anomalies.

2. Indexes:
   - Create indexes on frequently queried columns (e.g., Donor_ID, Donation_Date, Blood_Type) to improve query performance.

3. Constraints:
   - Enforce referential integrity using foreign key constraints to maintain relational consistency.
   - Define check constraints to enforce data validation rules (e.g., Blood_Type must be valid).

4. Data Integrity:
   - Implement triggers or stored procedures to enforce business rules and maintain data integrity.
   - Use transactions to ensure atomicity, consistency, isolation, and durability (ACID properties) during data operations.

5. Backup and Recovery:
   - Establish regular backup procedures to safeguard data against loss or corruption.
   - Implement mechanisms for data recovery in case of system failures or disasters.

6. Security:

- Implement access controls to restrict unauthorized access to sensitive data.
- Encrypt sensitive data (e.g., medical history) to protect privacy and confidentiality.

7. Scalability:

- Design the database schema with scalability in mind to accommodate future growth in data volume and user load.
- Consider partitioning strategies and sharding techniques for horizontal scalability.

8. Auditing and Logging:

- Implement auditing and logging mechanisms to track database changes and user activities.
- Maintain audit trails for compliance, forensic analysis, and troubleshooting purposes.

By following these design principles and considerations, the blood donation management database can efficiently store and manage data related to donors, donations, blood inventory, recipients, and other associated entities, thereby facilitating effective blood Donation management processes.

3.2 ENTITY RELATIONSHIP DIAGRAM

An entity–relationship model is usually the result of systematic analysis to define and describe what is important to processes in an area of a business. An E-R model does not define the business processes; it only presents a business data schema in graphical form. It is usually drawn in a graphical form as boxes (entities) that are connected by lines (relationships) which express the associations and dependencies between entities. Entities may be characterized not only by relationships, but also by additional properties (attributes), which include identifiers called "primary keys". Diagrams created to represent attributes as well as entities and relationships may be called entity-attribute-relationship diagrams, rather than entity-relationship models. An ER model is typically implemented as a database. In a simple relational database implementation, each row of a table represents one instance of an entity type, and each field in a table represents an attribute type. In a relational database a relationship between entities is implemented by storing the primary key of one entity as a pointer or "foreign key" in the table of another

entity. There is a tradition for ER/data models to be built at two or three levels of abstraction. Note that the conceptual-logical-physical hierarchy below is used in other kinds of specification, and is different from the three schema approach to software engineering. While useful for organizing data that can be represented by a relational structure, an entity-relationship diagram can't sufficiently represent semi-structured or unstructured data, and an ER Diagram is unlikely to be helpful on its own in integrating data into a pre-existing information system. Cardinality notations define the attributes of the relationship between the entities. Cardinalities can denote that an entity is optional.

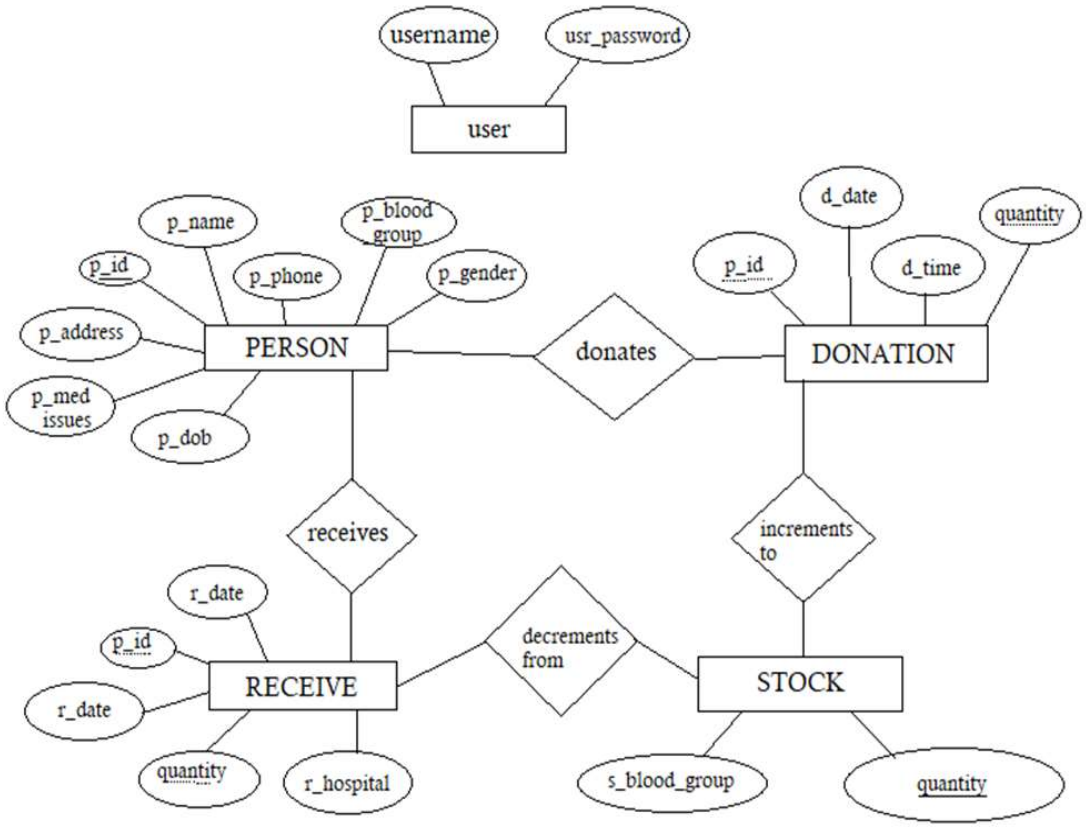

Fig. 3.2: ER diagram of Flights and Aircraft Database management system

3.3 RELATIONAL SCHEMA

The term "schema" refers to the organization of data as a blueprint of how the database is constructed. The formal definition of a database schema is a set of formulas called integrity constraints imposed on a database. A relational schema shows references among fields in the database. When a primary key is referenced

in another table in the database, it is called a foreign key. This is denoted by an arrow with the head pointing at the referenced key attribute. A schema diagram helps organize values in the database. The following diagram shows the schema diagram for the database.

3.4 DESCRIPTION OF TABLES

Sure, here's a basic description of tables for a blood donation database management system:

1. Donor Table:
   - Attributes: DonorID (Primary Key), Name, Blood Type, Gender, Age, Contact Information, Last Donation Date, Medical History.
2. Blood Bank Table:
   - Attributes: Bank ID (Primary Key), Name, Location, Contact Information, Available Blood Types, Capacity, Last Stock Update.
3. Blood Donation Table:
   - Attributes: Donation ID (Primary Key), Donor ID (Foreign Key), Donation Date, Blood Type, Quantity Donated, Blood Bank ID (Foreign Key).
4. Recipient Table:
   - Attributes: RecipientID (Primary Key), Name, Blood Type, Gender, Age, Contact Information, Medical History, Doctor's Information.
5. Blood Transfer Table:
   - Attributes: Transfer ID (Primary Key), From Blood Bank ID (Foreign Key), To Blood Bank ID (Foreign Key), Blood Type, Quantity Transferred, Transfer Date.
6. Staff Table:
   - Attributes: Staff ID (Primary Key), Name, Role, Contact Information, Blood BankID (Foreign Key).

These tables provide a foundation for managing donors, blood banks, donations, recipients, blood transfers, and staff within a blood donation system.

CHAPTER 4

IMPLEMENTATION

4.1 Login: Login for the new user

```
<!DOCTYPE html>
<html lang="en">
<?php
session start();
include('./db_connect.php');
ob_start();
if(!isset($_SESSION['system'])){
    $system = $conn->query("SELECT * FROM system_settings limit 1")->fetch_array();
    foreach($system as $k => $v){
        $_SESSION['system'][$k] = $v;
    }
}
ob_end_flush();
?>
<head>
  <meta charset="utf-8">
  <meta content="width=device-width, initial-scale=1.0" name="viewport">

  <title><?php echo $_SESSION['system']['name'] ?></title>


<?php include('./header.php'); ?>
<?php
if(isset($_SESSION['login_id']))
header("location:index.php?page=home");
?>
</head>
<style>
    body{
        width: 100%;
        height: calc(100%);
        /*background: #007bff;*/
    }
    main#main{
        width:100%;
        height: calc(100%);
        background:white;
    }
    #login-right{
        position: absolute;
        right:0;
        width:40%;
```

```
        height: calc(100%);
        background:white;
        display: flex;
        align-items: center;
    }
    #login-left{
        position: absolute;
        left:0;
        width:60%;
        height: calc(100%);
        background:#59b6ec61;
        display: flex;
        align-items: center;
        background: url(assets/uploads/blood-cells.jpg);
        background-repeat: no-repeat;
        background-size: cover;
    }
    #login-right .card{
        margin: auto;
        z-index: 1
    }
    .logo {
    margin: auto;
    font-size: 8rem;
    background: white;
    padding: .5em 0.7em;
    border-radius: 50% 50%;
    color: #000000b3;
    z-index: 10;
}
div#login-right::before {
    content: "";
    position: absolute;
    top: 0;
    left: 0;
    width: calc(100%);
    height: calc(100%);
    /*background: #000000e0;*/
}

</style>

<body>


  <main id="main" class=" bg-danger">
        <div id="login-left">
        </div>
```

```
      <div id="login-right" class="bg-danger">
         <div class="w-100">
         <h4 class="text-white text-center"><b><?php echo $_SESSION['system']['name']
?></b></h4>
         <br>
         <br>
         <div class="card col-md-8">
            <div class="card-body">
               <form id="login-form" >
                  <div class="form-group">
                     <label for="username" class="control-label">Username</label>
                     <input type="text" id="username" name="username" class="form-control">
                  </div>
                  <div class="form-group">
                     <label for="password" class="control-label">Password</label>
                     <input type="password" id="password" name="password" class="form-control">
                  </div>
                  <center><button class="btn-sm btn-block btn-wave col-md-4 btn-
primary">Login</button></center>
               </form>
            </div>
         </div>
         </div>
      </div>


  </main>

  <a href="#" class="back-to-top"><i class="icofont-simple-up"></i></a>


</body>
<script>
   $('#login-form').submit(function(e){
      e.preventDefault()
      $('#login-form button[type="button"]').attr('disabled',true).html('Logging in...');
      if($(this).find('.alert-danger').length > 0 )
         $(this).find('.alert-danger').remove();
      $.ajax({
         url:'ajax.php?action=login',
         method:'POST',
         data:$(this).serialize(),
         error:err=>{
            console.log(err)
      $('#login-form button[type="button"]').removeAttr('disabled').html('Login');

         },
```

```
        success:function(resp){
            if(resp == 1){
                location.href ='index.php?page=home';
            }else{
                $('#login-form').prepend('<div class="alert alert-danger">Username or password is incorrect.</div>')
                $('#login-form button[type="button"]').removeAttr('disabled').html('Login');
            }
        }
    })
  })
</script>
</html>
```

4.2 BLOOD DONAR:

```
<?php
include 'db_connect.php';
if(isset($_GET['id'])){
$qry = $conn->query("SELECT * FROM blood_inventory where id= ".$_GET['id']);
foreach($qry->fetch_array() as $k => $val){
  $$k=$val;
}
}
?>
<div class="container-fluid">
  <form action="" id="manage-donation">
    <input type="hidden" name="id" value="<?php echo isset($id) ? $id : '' ?>">
    <input type="hidden" name="status" value="1">
    <div id="msg"></div>
    <div class="form-group">
      <label for="" class="control-label">Donor's Name</label>
      <select class="custom-select select2" name="donor_id" required>
        <option value=""></option>
        <?php
        $qry = $conn->query("SELECT * FROM donors order by name asc");
        while($row= $qry->fetch_assoc()):
        ?>
        <option value="<?php echo $row['id'] ?>" data-bgroup="<?php echo $row['blood_group'] ?>" <?php echo isset($donor_id) && $donor_id == $row['id'] ? 'selected' : '' ?>><?php echo $row['name'] ?></option>
        <?php endwhile; ?>
      </select>
    </div>
    <div class="form-group">
      <label for="" class="control-label">Blood Group</label>
```

```
        <input type="text" class="form-control" name="blood_group" value="<?php echo isset($blood_group) ? $blood_group :" ?>" required readonly>

      </div>
      <div class="form-group">
        <label for="" class="control-label">Volume (ml)</label>
        <input type="number" class="form-control text-right" step="any" name="volume" value="<?php echo isset($volume) ? $volume :" ?>" required>
      </div>
      <div class="form-group">
        <label for="" class="control-label">Date of Transfusion/Donation</label>
        <input type="date" class="form-control" name="date_created" value="<?php echo isset($date_created) ? date('Y-m-d',strtotime($date_created)) :" ?>" required>
      </div>
    </form>
</div>
<script>
  $('.select2').select2({
    placeholder:'Please select here.',
    width:'100%'
  })
  $(document).ready(function(){
    if('<?php echo isset($donor_id)? 1:0 ?>' == 1)
      $('[name="donor_id"]').trigger('change')
  })


  $('[name="donor_id"]').change(function(){
    var _id = $(this).val()
    if(_id > 0){
      $('[name="blood_group"]').val($(this).find('option[value="'+_id+'"]').attr('data-bgroup'))
    }else{
      $('[name="blood_group"]').val('')
    }
  })
  $('#manage-donation').submit(function(e){
    e.preventDefault()
    start_load()
    $('#msg').html('')
    $.ajax({
      url:'ajax.php?action=save_donation',
      data: new FormData($(this)[0]),
      cache: false,
      contentType: false,
      processData: false,
      method: 'POST',
      type: 'POST',
      success:function(resp){
```

```
            if(resp==1){
                alert_toast("Data successfully saved.",'success')
                    setTimeout(function(){
                        location.reload()
                    },1000)
            }
        }
    })
  })
</script>
```

4.2 LIST OF DONATION

```
<?php
include 'db_connect.php';
if(isset($_GET['id'])){
$qry = $conn->query("SELECT * FROM donors where id= ".$_GET['id']);
foreach($qry->fetch_array() as $k => $val){
   $$k=$val;
}
}
?>
<div class="container-fluid">
   <form action="" id="manage-donor">
      <input type="hidden" name="id" value="<?php echo isset($id) ? $id : '' ?>">
      <div class="form-group">
         <label for="" class="control-label">Full Name</label>
         <input type="text" class="form-control" name="name"  value="<?php echo isset($name) ?
$name :'' ?>" required>
      </div>
      <div class="form-group">
         <label for="" class="control-label">Address</label>
         <textarea cols="30" rows = "2" required="" name="address" class="form-control"><?php
echo isset($address) ? $address :'' ?></textarea>
      </div>
      <div class="form-group">
         <label for="" class="control-label">Email</label>
         <input type="email" class="form-control" name="email"  value="<?php echo isset($email)
? $email :'' ?>" required>
      </div>
      <div class="form-group">
         <label for="" class="control-label">Contact #</label>
         <input type="text" class="form-control" name="contact" value="<?php echo
isset($contact) ? $contact :'' ?>" required>
      </div>
      <div class="form-group">
         <label for="" class="control-label">Blood Group</label>
```

```
        <select name="blood_group" id="" class="custom-select select2" required>
          <option <?php echo isset($blood_group) && $blood_group == 'A+' ? ' selected' : '' ?>>A+</option>
          <option <?php echo isset($blood_group) && $blood_group == 'B+' ? ' selected' : '' ?>>B+</option>
          <option <?php echo isset($blood_group) && $blood_group == 'O+' ? ' selected' : '' ?>>O+</option>
          <option <?php echo isset($blood_group) && $blood_group == 'AB+' ? ' selected' : '' ?>>AB+</option>
          <option <?php echo isset($blood_group) && $blood_group == 'A-' ? ' selected' : '' ?>>A-</option>
          <option <?php echo isset($blood_group) && $blood_group == 'B-' ? ' selected' : '' ?>>B-</option>
          <option <?php echo isset($blood_group) && $blood_group == 'O-' ? ' selected' : '' ?>>O-</option>
          <option <?php echo isset($blood_group) && $blood_group == 'AB-' ? ' selected' : '' ?>>AB-</option>
        </select>
      </div>
    </form>
</div>
<script>

    $('#manage-donor').submit(function(e){
        e.preventDefault()
        start_load()
        $('#msg').html('')
        $.ajax({
            url:'ajax.php?action=save_donor',
            data: new FormData($(this)[0]),
            cache: false,
            contentType: false,
            processData: false,
            method: 'POST',
            type: 'POST',
            success:function(resp){
                if(resp==1){
                    alert_toast("Data successfully saved.",'success')
                        setTimeout(function(){
                            location.reload()
                        },1000)
                }
            }
        })
    })
</script>
```

4.3 SUBJECT ENTRY

```
<?php include 'db_connect.php' ?>
<style>
  span.float-right.summary_icon {
   font-size: 3rem;
   position: absolute;
   right: 1rem;
   top: 0;
}
.imgs{
                margin: .5em;
                max-width: calc(100%);
                max-height: calc(100%);
        }
        .imgs img{
                max-width: calc(100%);
                max-height: calc(100%);
                cursor: pointer;
        }
        #imagesCarousel,#imagesCarousel .carousel-inner,#imagesCarousel .carousel-item{
                height: 60vh !important;background: black;
        }
        #imagesCarousel .carousel-item.active{
                display: flex !important;
        }
        #imagesCarousel .carousel-item-next{
                display: flex !important;
        }
        #imagesCarousel .carousel-item img{
                margin: auto;
        }
        #imagesCarousel img{
                width: auto!important;
                height: auto!important;
                max-height: calc(100%)!important;
                max-width: calc(100%)!important;
        }
</style>

<div class="containe-fluid">
        <div class="row mt-3 ml-3 mr-3">
      <div class="col-lg-12">
         <div class="card">
           <div class="card-body">
             <?php echo "Welcome back ". $_SESSION['login_name']."!"  ?>
             <hr>
```

```
<h4><b>Available Blood per group in Liters</b></h4>
<div class="row">
  <?php
  $blood_group = array("A+","B+","O+","AB+","A-","B-","O-","AB-");
  foreach($blood_group as $v){
    $bg_inn[$v] = 0;
    $bg_out[$v] = 0;
  }
  $qry = $conn->query("SELECT * FROM blood_inventory ");
  while($row = $qry->fetch_assoc()){
    if($row['status'] == 1){
      $bg_inn[$row['blood_group']] += $row['volume'];
    }else{
      $bg_out[$row['blood_group']] += $row['volume'];
    }
  }

  ?>
  <?php foreach ($blood_group as $v): ?>
  <div class="col-md-3 mb-3">
    <div class="card">
      <div class="card-body bg-light">
        <div class="card-body text-dark">
          <span class="float-right summary_icon"> <?php echo $v ?> <i class="fa
fa-tint text-danger"></i></span>
          <h4><b>
            <?php echo ($bg_inn[$v] - $bg_out[$v]) / 1000 ?>
          </b></h4>
        </div>
      </div>
    </div>
  </div>
<?php endforeach; ?>
</div>
<hr>
<div class="row">
  <div class="col-md-3 mb-3">
    <div class="card">
      <div class="card-body bg-light">
        <div class="card-body text-dark">
          <span class="float-right summary_icon"> <i class="fa fa-user-friends
text-primary "></i></span>
          <h4><b>
            <?php echo $conn->query("SELECT * FROM donors")->num_rows
?>
          </b></h4>
          <p><b>Total Donors</b></p>
        </div>
```

```
                    </div>
                  </div>
                </div>
                <div class="col-md-3 mb-3">
                  <div class="card">
                    <div class="card-body bg-light">
                      <div class="card-body text-dark">
                        <span class="float-right summary_icon"> <i class="fa fa-notes-medical text-danger "></i></span>
                        <h4><b>
                          <?php echo $conn->query("SELECT * FROM blood_inventory where status = 1 and date(date_created) = '".date('Y-m-d')."' ")->num_rows ?>
                        </b></h4>
                        <p><b>Total Donated Today</b></p>
                      </div>
                    </div>
                  </div>
                </div>
                <div class="col-md-3 mb-3">
                  <div class="card">
                    <div class="card-body bg-light">
                      <div class="card-body text-dark">
                        <span class="float-right summary_icon"> <i class="fa fa-th-list "></i></span>
                        <h4><b>
                          <?php echo $conn->query("SELECT * FROM requests where date(date_created) = '".date('Y-m-d')."' ")->num_rows ?>
                        </b></h4>
                        <p><b>Today's Requests</b></p>
                      </div>
                    </div>
                  </div>
                </div>
                <div class="col-md-3 mb-3">
                  <div class="card">
                    <div class="card-body bg-light">
                      <div class="card-body text-dark">
                        <span class="float-right summary_icon"> <i class="fa fa-check text-primary "></i></span>
                        <h4><b>
                          <?php echo $conn->query("SELECT * FROM requests where date(date_created) = '".date('Y-m-d')."' and status = 1 ")->num_rows ?>
                        </b></h4>
                        <p><b>Today's Approved Requests</b></p>
                      </div>
                    </div>
                  </div>
                </div>
```

```
                </div>
            </div>
        </div>
    </div>
</div>
<script>
        $('#manage-records').submit(function(e){
    e.preventDefault()
    start_load()
    $.ajax({
        url:'ajax.php?action=save_track',
        data: new FormData($(this)[0]),
        cache: false,
        contentType: false,
        processData: false,
        method: 'POST',
        type: 'POST',
        success:function(resp){
            resp=JSON.parse(resp)
            if(resp.status==1){
                alert_toast("Data successfully saved",'success')
                setTimeout(function(){
                    location.reload()
                },800)

            }

        }
    })
})
$('#tracking_id').on('keypress',function(e){
    if(e.which == 13){
        get_person()
    }
})
$('#check').on('click',function(e){
        get_person()
})
function get_person(){
        start_load()
    $.ajax({
            url:'ajax.php?action=get_pdetails',
            method:"POST",
            data:{tracking_id : $('#tracking_id').val()},
            success:function(resp){
                if(resp){
                    resp = JSON.parse(resp)
```

```
                if(resp.status == 1){
                    $('#name').html(resp.name)
                    $('#address').html(resp.address)
                    $('[name="person_id"]').val(resp.id)
                    $('#details').show()
                    end_load()

                }else if(resp.status == 2){
                    alert_toast("Unknow tracking id.",'danger');
                    end_load();
                }
            }
        }
    })
  }
</script>
```

4.4 RESULT

The implementation of the modules, as outlined above, aims to provide a structured and organized approach to the development of the Online Blood Donation Management System. The use of PHP, in combination with HTML, CSS, Bootstrap, Ajax, and JQuery, facilitates the creation of a dynamic and responsive system with distinct functionalities for both administrators and users.The systematic deployment of these modules enhances the user experience, enables efficient management of blood donation, and ensures seamless interaction with the system. The clear separation of functionalities into modular components also promotes maintainability and scalability of the project.

CHAPTER 5

TESTING

5.1 SOFTWARE TESTING

Testing for a blood management system would involve various types of software testing to ensure its reliability, accuracy, and security. Here are some key types of testing for a blood management system:

1. Functional Testing:
   - Verify that all functionalities of the system work as expected, such as donor registration, blood donation recording, inventory management, recipient matching, etc.
2. Integration Testing:
   - Ensure that different modules of the system integrate seamlessly, such as the donor module with the inventory module, or the blood bank module with the recipient module.
3. Usability Testing:
   - Evaluate the user interface for ease of use, intuitiveness, and accessibility. Ensure that users can navigate through the system efficiently.
4. Performance Testing:
   - Test the system's performance under various load conditions to ensure it can handle multiple concurrent users and transactions without significant slowdowns or crashes.
5. Security Testing:
   - Assess the system's security measures to protect sensitive information such as donor details and medical records. This includes testing for vulnerabilities like SQL injection, data breaches, and unauthorized access.
6. Compatibility Testing:
   - Verify that the system works correctly across different devices, browsers, and operating systems to ensure a consistent user experience.
7. Regression Testing:
   - Ensure that new updates or changes to the system do not introduce bugs or regressions in existing functionalities.
8. User Acceptance Testing (UAT):
   - Involve end-users in testing to gather feedback and ensure that the system meets their requirements and expectations.
9. Data Integrity Testing:

- Verify the accuracy and consistency of data stored in the system, such as donor information, blood inventory levels, and recipient records.

10. Disaster Recovery Testing:

- Test the system's ability to recover from disasters or unexpected failures, such as database crashes or server outages, to minimize downtime and data loss.

By conducting thorough testing across these different areas, you can ensure that the blood management system is robust, reliable, and secure for its intended users.

5.2 MODULE TESTING AND INTEGRATION

Module Testing: Each module of the system, such as the database management module and the user interface module, will be tested individually to ensure they function correctly. Integration Testing: Once individual modules are tested, they will be integrated and tested together to ensure they work together seamlessly.

5.3 LIMITATIONS

Several limitations may be encountered in a blood management system:

1. Data Accuracy and Integrity: Despite efforts to maintain accurate records, human error or technical issues could lead to inaccuracies in donor information, blood inventory levels, or recipient records. Regular audits and data validation checks can help mitigate this issue.
2. Dependency on Donor Availability: The system relies heavily on the availability of voluntary donors. During periods of low donor turnout, blood shortages may occur, impacting the system's ability to meet the demand for blood products.
3. Storage and Shelf Life: Blood products have limited shelf lives, and proper storage conditions are crucial to maintaining their viability. Storage errors or inadequate facilities could lead to wastage of blood products, reducing the effectiveness of the system.
4. Geographical Constraints: Rural or remote areas may have limited access to blood banks or donation centers, resulting in challenges in supplying blood products to those regions. Transportation logistics and infrastructure limitations can further exacerbate this issue.
5. Blood Type Availability: Not all blood types are equally prevalent in the population, and certain rare blood types may be in short supply. Matching donors with compatible recipients can be challenging for rare blood types, potentially delaying treatment for patients.
6. Regulatory Compliance: Blood management systems must adhere to strict regulatory standards and guidelines to ensure the safety and quality of blood products. Non-

compliance with regulatory requirements can lead to legal issues and compromise patient safety.

7. Technological Limitations: Outdated or inefficient technology infrastructure may hinder the system's performance and scalability. Upgrading systems and implementing newer technologies can be costly and time-consuming.

8. Staffing and Training: Adequate staffing and ongoing training are essential for the effective operation of a blood management system. Shortages of qualified personnel or inadequate training can impact the system's efficiency and reliability.

9. Emergency Response Preparedness: The system must be prepared to respond effectively to emergencies such as natural disasters or mass casualty incidents. Ensuring sufficient blood supplies and rapid deployment of resources during emergencies can be challenging.

10. Ethical Considerations: Blood donation and transfusion involve ethical considerations such as informed consent, confidentiality, and equitable distribution of blood products. Ethical dilemmas may arise in situations where demand exceeds supply or when prioritizing patients for treatment.

Addressing these limitations requires a multi-faceted approach, including continuous quality improvement, stakeholder collaboration, investment in technology and infrastructure, and adherence to ethical and regulatory standards.

CHAPTER 6

SNAPSHOTS

6.1 LOGIN PAGE

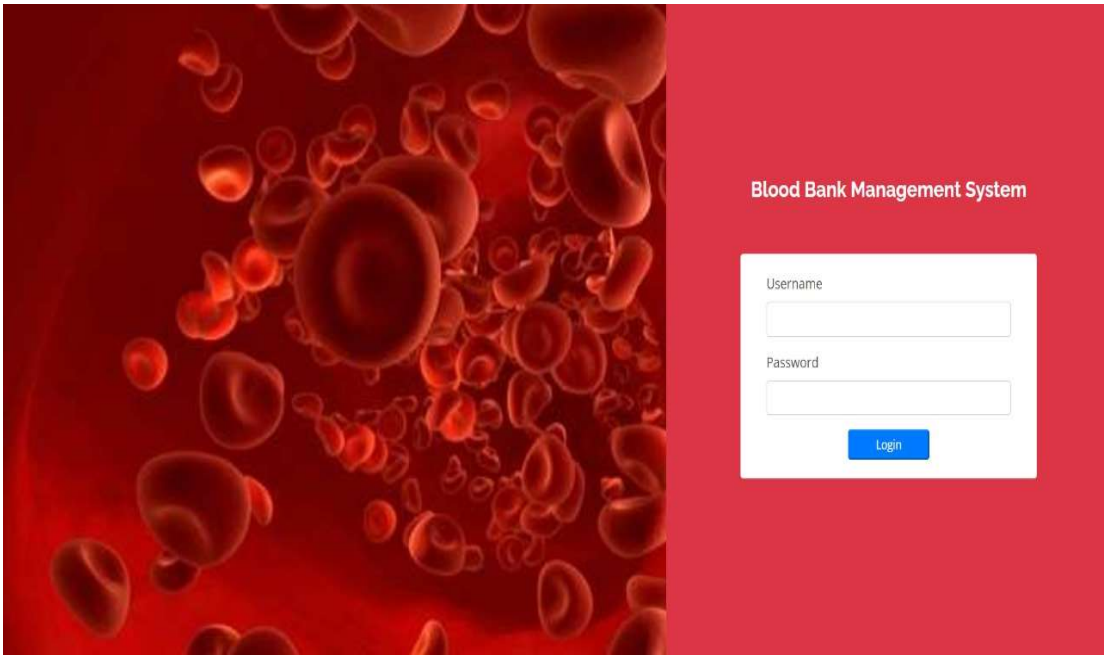

Fig 6.1 login page

6.2 HOME PAGE

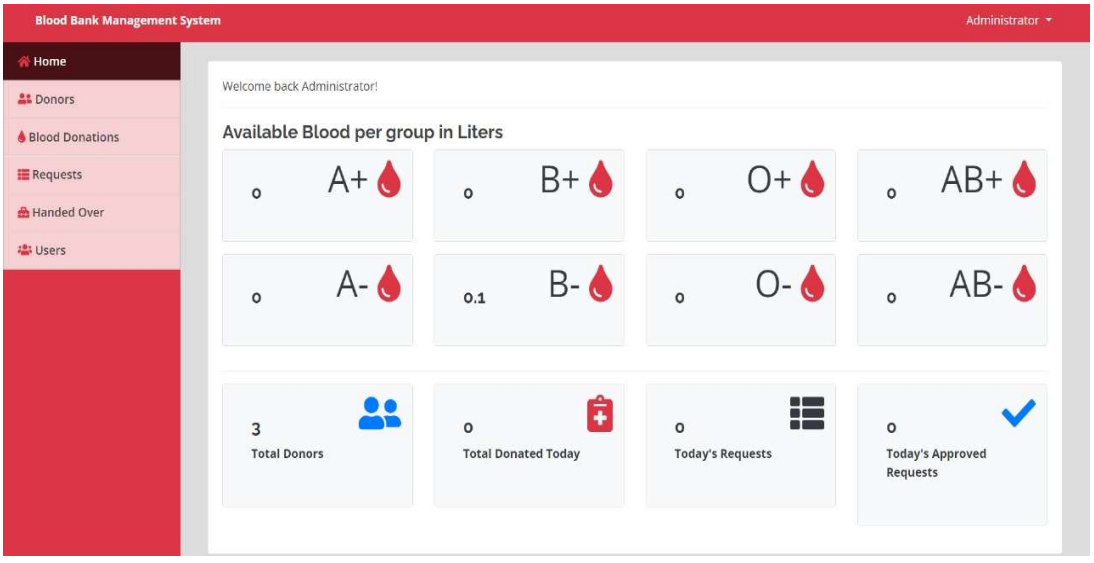

Fig 6.2 Home Page

6.3 DONOR ENTRY

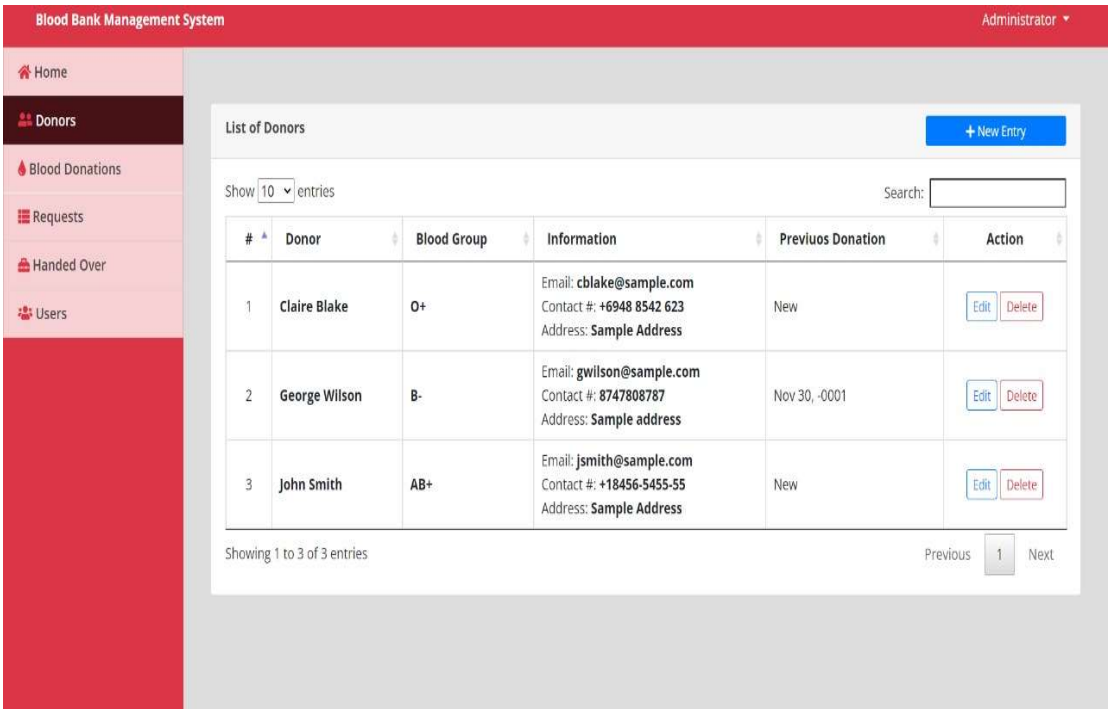

Fig 6.4 Donor Entry

6.4 LIST OF DONATIONS

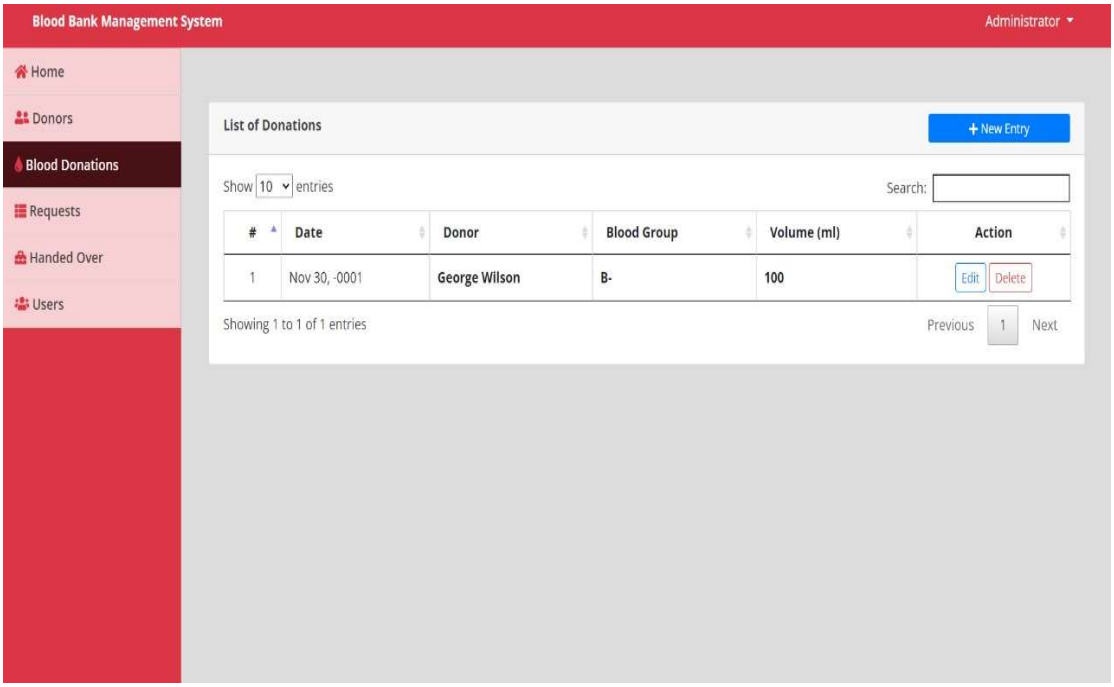

Fig 6.5:List of donations

CHAPTER 7

CONCLUSION

The Blood Bank Management System is a web-based application that manages the process of blood donation from registration to distribution. The system uses a Database Management System (DBMS) and Java Database Connectivity (JDBC) technology to store and manage donor information, blood types, and inventory records. The system provides an easy-to-use interface for managing blood donations, scheduling appointments, and monitoring inventory levels. Overall, The Blood Bank Management System is an essential tool for managing the blood donation process and improving the efficiency and effectiveness of blood banks and hospitals. The system can be The Blood Bank Management System is a based application that manages the process of blood donation from registration to distribution. The system uses a Database Management System (DBMS) and Java Database Connectivity (JDBC) technology ge donor information, blood types, and inventory records. The use interface for managing blood donations, scheduling appointments, and monitoring inventory The Blood Bank Management System is load donation process is efficient, safe, and meets the needs of hospitals and patients. The system enables blood banks to manage donor information and blood inventory records efficiently, and it enables hospitals to request and receive blood donations. The system can be proved by adding new features like integration with other systems, mobile application, analytics and reporting, online blood bank, and integration with IoT devices. These improvements can help in the efficient utilization of blood samples, reduce wastage, and ultimately save more lives.

CHAPTER 8

FUTURE ENHANCEMENTS

The blood bank management system has the potential for further development and improvement. Some future aspects of the project include: 1)Integration with other systems: The blood bank management system can be integrated with other hospital management systems to provide a seamless experience for the hospital staff. This integration can help in the efficient sharing of data between different systems, reducing manual errors and improving overall efficiency.

2)Mobile Application: A mobile application can be developed for the blood bank management system. This application can allow donors to register and make appointments for blood donations. The application can also allow for tracking of the donor's blood donation history and provide notifications when their blood type is in high demand.

3)Analytics and Reporting: The blood bank management system can be improved by adding analytics and reporting capabilities. The system can generate reports on blood donation trends, inventory levels, and other relevant information. These reports can help blood banks in making informed decisions about blood donations and transfusions.

4)Integration with IoT Devices: IoT devices can be integrated with the blood bank management system to monitor the temperature and humidity levels of blood samples. This can help in ensuring that the blood samples are stored at the appropriate temperature and avoid spoilage.

5)Donor incentives program: The system can be enhanced to include a donor incentives program that rewards donors for their contributions. This would encourage more people to donate blood and help increase the overall blood supply.

REFERENCES

1. Tuan, T. G. Online Blood Donation Reservation and Management System, 2006.

2. Development of a Blood Bank Management System Sumazly Sulaimana,*, Abdul Aziz K.Abdul Hamida , Nurul Ain Najihah Yusria.

3. Vikas Kulshreshtha, Dr. Sharad Maheshwari, “Blood Bank Management Information System in India,” in International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622, Vol. 1, Issue 2, pp.260-263

4. Ravi Kumar, Shubham Singh, V Anu Ragavi, “Blood Bank Management System,” IJARIIE-ISSN(O)- 2395- 4396, Vol-3 Issue-5 2017

5. Blood donor selection. Guidelines on assessing donor suitability for blood donation. Annex 3. Geneva: World Health Organization; 2012. [17 August 2012]. http://www .who.int/bloodsafety/voluntary_donation/blood_donor _selection_counselling/en/ [PubMed]

6. H.Lowalekar and N. Ravicharan, "Blood Bank inventory management in India",OPSEARCH,vol.51, no. 3,pp. 376-399,2014

7. Ekanayaka, E. M. S. S., & Wimaladharma, C. (2015). Blood bank management system