CHAPTER 1

INTRODUCTION

1.1 Overview

"Blood bank management system" is designed to develop an online blood system applying the concepts of database security and encryption that is easily retrievable. The blood bank management is known as a pilot project that is designed for the blood bank to gather blood from various sources and distribute to the needy people who have requirement. The software is designed to handle the daily transactions of blood and search the details when required. It helps to register the details of donors, blood collection details as well as blood issued reports.

1.2 Problem Statement

The main aim of "blood bank management system" is Admin/User has to login first. Any person who is willing to donate/receive blood will have to register by giving all the personal details. Admin should register the given details. The admin will be able to view all the details and records of donors as well as receivers and the stock of the blood bank. We will be using the concepts of database encryption to make sure that user information is to kept secure and confidential. This will help to keep donation records protected from any threats from individuals with potentially malicious intensions or hazards to the security of the data.

1.3 Database Management System

A Database Management System (DBMS) is system software designed to create, organize, and manage databases efficiently. It acts as an intermediary between databases and application programs, facilitating data retrieval, modification, and organization. The DBMS oversees data management, database engine operations for access and modification, and the database schema, which defines its logical structure. These components ensure concurrency, data integrity, and consistent administration. DBMS supports key tasks such as change management, performance monitoring, and backup, while also automating rollbacks, recovery, and activity logging and auditing.

1.4 SQL

SQL (Structured Query Language) is a standard programming language utilized for managing and manipulating relational databases. It provides a comprehensive set of commands for querying data, inserting, updating, and deleting records, as well as defining and modifying the structure of databases. With SQL, users can execute various tasks including retrieving specific data using SELECT statements, defining constraints to ensure data integrity, controlling access permissions, and managing transactions to maintain data consistency. SQL is widely adopted across different industries and applications for its versatility in handling data management tasks efficiently and reliably.

1.5 HTML

HTML, or Hypertext Markup Language, is the standard language used to create and design web pages. It consists of a series of elements or tags that define the structure and content of a webpage. Each element serves a specific purpose, such as headings, paragraphs, links, images, and forms. HTML documents are composed of nested elements, forming a hierarchical structure that browsers interpret to render web content. With its simplicity and versatility, HTML remains the backbone of the World Wide Web, providing the foundation for creating rich and interactive digital experiences on the internet.

1.6 CSS

CSS, or Cascading Style Sheets, is a stylesheet language used to enhance the presentation and layout of HTML documents. With CSS, web designers can control the appearance of elements on a webpage, including colors, fonts, spacing, and positioning. By separating the content from its presentation, CSS enables consistent styling across multiple pages and ensures a seamless user experience. CSS operates through selectors that target specific HTML elements, allowing designers to apply styles selectively. Through cascading rules and inheritance, CSS facilitates the creation of responsive and visually appealing websites tailored to various devices and screen sizes. It plays a crucial role in shaping the aesthetic and user interface of modern web applications, contributing to the overall look and feel of the digital landscape.

1.7 Javascript

JavaScript, often abbreviated as JS, is a dynamic programming language commonly used for web development. It runs directly in the web browser, enabling developers to create interactive and responsive features on web pages. JavaScript is versatile, allowing developers to manipulate the content and behaviour of HTML and CSS dynamically. With JavaScript, one can create functionalities such as form validation, image sliders, interactive maps, and much more. It also facilitates event handling, enabling developers to respond to user actions like clicks, mouse movements, and keystrokes. JavaScript's simplicity and ubiquity make it an essential tool for enhancing the user experience of websites, providing dynamic and engaging content without the need for server-side processing.

1.8 PHP

PHP, which stands for Hypertext Pre-processor, is a server-side scripting language commonly used for web development. It is embedded within HTML code and executed on the server, generating dynamic content that is then sent to the client's web browser. PHP is highly versatile and widely supported, making it a popular choice for developing dynamic websites and web applications.

PHP can perform various tasks, such as processing form data, interacting with databases, generating dynamic page content, handling cookies and sessions, and performing file operations. It is often used in conjunction with databases like MySQL to create powerful web applications that can store and retrieve data dynamically.

One of the key advantages of PHP is its ease of use and accessibility, as it integrates seamlessly with HTML and requires minimal setup to get started. Additionally, PHP is open-source and has a large community of developers, which means there are extensive libraries, frameworks, and resources available to assist developers in building robust and scalable web applications.

1.9 SQL Connections

To establish a connection between PHP and a MySQL database, the mysqli class is commonly employed. mysqli is a PHP extension that provides enhanced functionality for interacting with MySQL databases, including features such as prepared statements and transactions.

CHAPTER 2

REQUIREMENTS SPECIFICATION

2.1 Software Requirements:

- XAMPP (includes Apache HTTP Server, MySQL, PHP)
- Database Support - MySQL 5.7
- MySQL Workbench
- Operating System – Windows 7 / Ubuntu 16.04
- Web Browser – Firefox 50 or later, Google Chrome 60 or later

2.2 Hardware Requirements:

- Processor – Intel Core i3 or above
- RAM – 2 GB or more
- Hard Disk – 3 GB or more
- Monitor – VGA of 1024x768 screen resolution

CHAPTER 3

DETAILED DESIGN

3.1 System Design

The system follows a client-server architecture, where the client-side consists of web browsers (Firefox, Chrome) accessing the academic dashboard through HTTP requests sent to the server. The server-side comprises the XAMPP stack running on the server machine, including Apache HTTP Server for handling web requests, MySQL for managing the database, and PHP for server-side scripting.

Components:

Client-Side:

Web Browsers (Firefox, Chrome): Users access the academic dashboard through web browsers. They interact with the frontend interface developed using HTML/CSS.

Server-Side:

Apache HTTP Server: Serves as the web server to handle client requests. It processes HTTP requests and responds with dynamic content generated by PHP scripts.

MySQL Database: Manages the relational database to store student and teacher data, marks, attendance records, and alerts. Utilized for data storage and retrieval.

PHP: Handles server-side scripting to interact with the database, process user requests, and generate dynamic web content. PHP scripts are responsible for fetching and updating data from the database based on user interactions.

Development Environment:

XAMPP: Provides an easy-to-install package containing Apache HTTP Server, MySQL, PHP, and Perl. It is used for local development and testing of the academic dashboard.

3.2 Entity Relationship Diagram

An Entity-Relationship (ER) diagram is a visual representation of the data model that shows the relationships between different entities in a system or database. It's a fundamental tool used in database design to illustrate the logical structure of the database and the relationships between various entities.

Key components of an ER diagram include:

Entities: Entities represent the real-world objects or concepts, such as a person, place, thing, or event, that are stored in the database. Each entity is depicted as a rectangle in the diagram.

Attributes: Attributes are the properties or characteristics of entities. They describe the features or qualities of an entity. Attributes are represented as ovals connected to their respective entity rectangles.

Relationships: Relationships depict the associations between entities. They describe how entities are connected or linked to each other within the database. Relationships are typically represented as lines connecting two entities, and they may include cardinality and participation constraints to specify the nature of the relationship.

Cardinality: Cardinality defines the number of instances of one entity that can be associated with another entity. It specifies the minimum and maximum number of occurrences of one entity that can be related to another entity.

Primary Key: A primary key is an attribute or a combination of attributes that uniquely identifies each instance of an entity in the database. It ensures the integrity and uniqueness of the data.

Foreign Key: A foreign key is an attribute in one entity that refers to the primary key of another entity. It establishes a relationship between the two entities by linking their respective records.

ER diagrams provide a clear and concise way to visualize the database schema, including its structure and relationships. They serve as a blueprint for database design and help database designers, developers, and stakeholders understand the data model and its requirements.

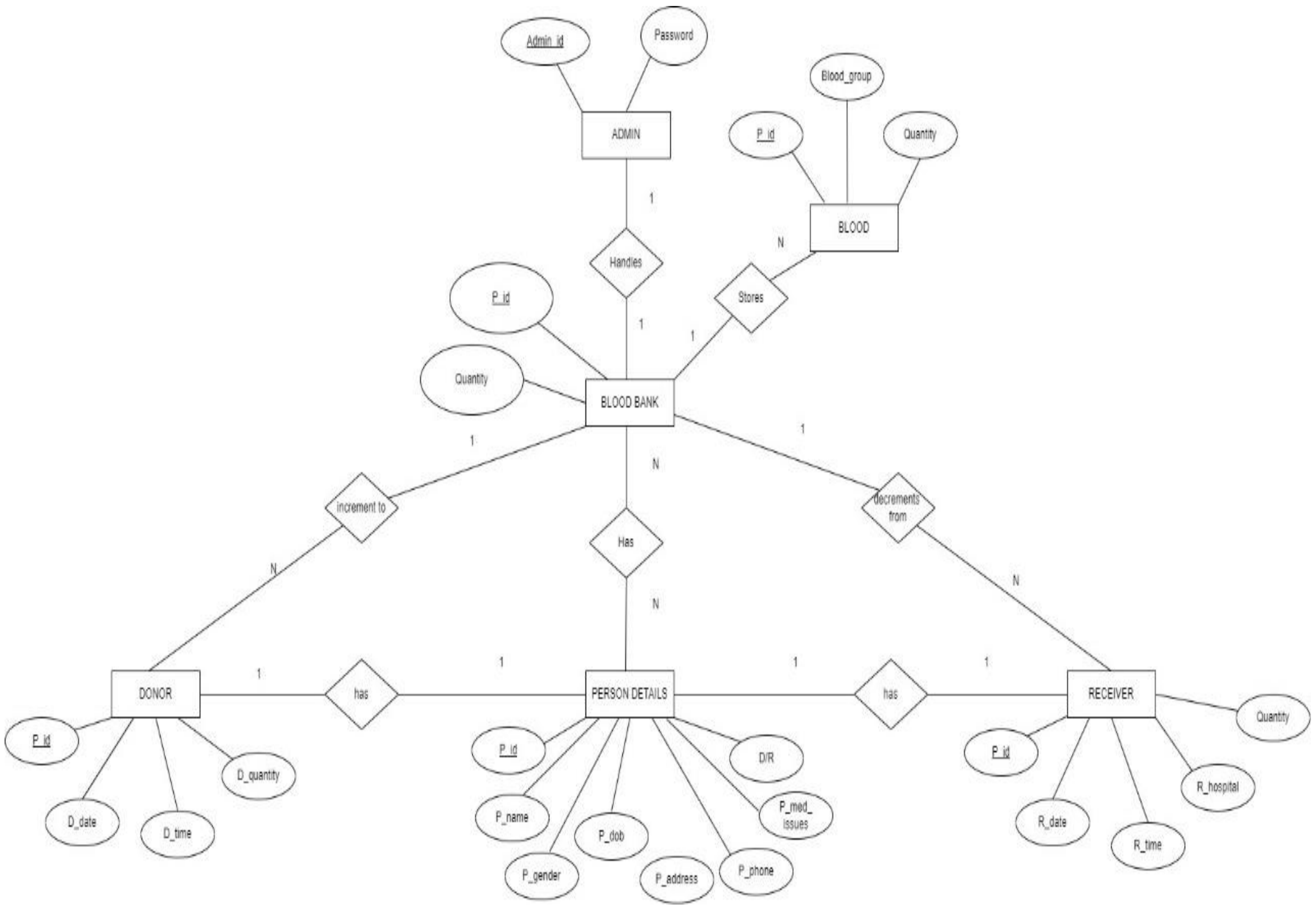


Fig 3.1 – ER Diagram of Blood Bank Management Sytem

3.2 Relational Schema

A schema diagram is a visual representation of the logical structure and relationships within a database. It provides a high-level overview of the database schema, including its entities (tables), attributes (columns), and the relationships between entities.

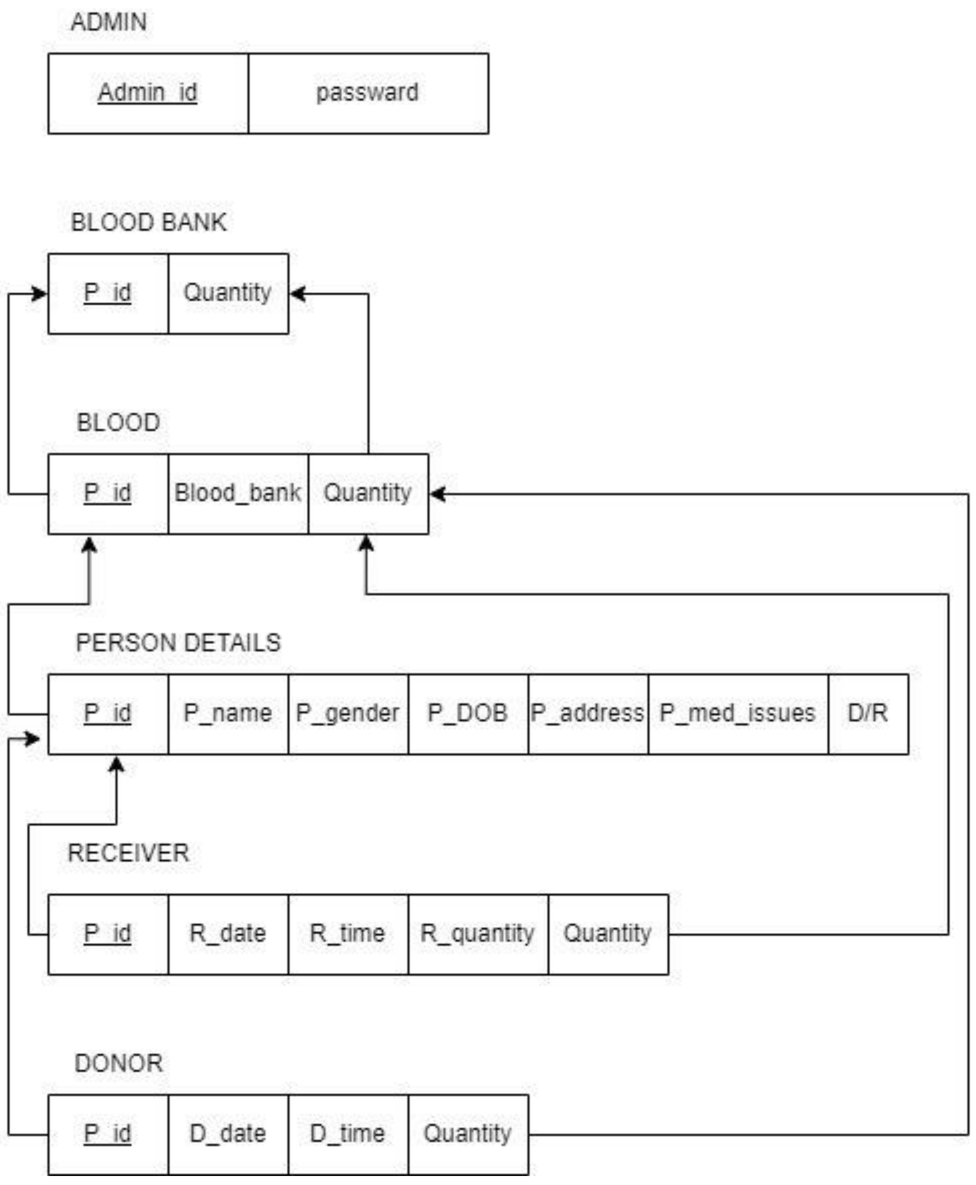


Fig 3.2 – Schema Diagram of Blood Bank Management Sytem

3.4 Database Tables

NAME	TYPE	FIELD NAME	DESCRIPTION
Admin_id	Varchar (10)	not null	id of employee
password	Varchar (16)	not null	password to login

TABLE 4.1: ADMIN

NAME	TYPE	FIELD NAME	DESCRIPTION
p_ id	Int (10)	not null	unique donor id (auto increment)
p_ name	Varchar (25)	not null	donor name
p_ phone	Char (10)	not null	phone number
P_ dob	date	not null	date of birth
P_ address	Varchar (100)	not null	address
p_ gender	Char (1)	not null	gender
p_ blood_ group	Varchar (3)	not null	blood group
p_ med_ issues	Varchar (100)	null	any medical issues to be mentioned

TABLE 4.2: PERSON

NAME	TYPE	FIELD NAME	DESCRIPTION
p_ id	Int (10)	not null	unique receiver id
r_ date	date	not null	blood donation date
r_ time	time	not null	blood donation time
r_name	Varchar (25)	not null	receiver name
quantity	Int (5)	null	Units of blood available

TABLE 4.3: RECEIVER

NAME	TYPE	FIELD NAME	DESCRIPTION
p_ id	Int (10)	not null	unique person id
r_ date	date	not null	date of blood received
r_ time	time	not null	time of blood received
quantity	Int (1)	not null	units of blood received (bottles)

TABLE 4.4: DONOR

NAME	TYPE	FIELD NAME	DESCRIPTION
p_id	Int (5)	not null	Unique person id
quantity	Int (5)	null	units of blood available

TABLE 4.5: BLOOD BANK

NAME	TYPE	FIELD NAME	DESCRIPTION
p_id	Int (5)	not null	Unique person id
quantity	Int (5)	null	units of blood available

TABLE 4.6: BLOOD

3.5 USER FLOW

5.5.1 Login page:

Step1: Enter the USERNAME in the given field.

Step2: Enter the valid PASSWORD.

Step3: Click on LOGIN button

5.5.2 Home page:

This page contains agenda of the database. Reason behind creating the database and what is the motive of this db.

It also gives the brief information about number of registrations, donations and receives taken place in the blood management.

5.5.3 Person page:

Here the user must enter the personal details of the new donor/receiver.

Step1: Open the ADD PERSON page.

Step2: The PERSON ID is auto incremented because its unique for all.

Step3: Enter the PERSON NAME.

Step4: Enter the PHONE NUMBER.

Step5: Enter the GENDER.

Step6: Select the DATE OF BIRTH.

Step7: Select the BLOOD GROUP.

Step8: Enter the ADDRESS.

Step9: Enter the MEDICAL ISSUES if any.

Step10: Click on REGISTER button.

5.5.4 Search Person Page:

In this page one can retrieve the details of the person using PERSON ID.

Step1: Open the SEARCH PERSON page

Step2: Enter the PERSON ID.

Step3: Click on SUBMIT button.

5.5.5 Donation page:

If any person is willing to donate the blood.

Step1: Open the NEW DONATION page.

Step2: Enter the unique PERSON ID of the person.

Step3: Enter the Units of blood donated.

Step4: Click on SUBMIT button.

(Note: Person details are accessed using the PERSON ID, date and time is automatically accessed using computers default time.)

5.5.6 Receive page:

If any person require blood for emergency purpose.

Step1: Open the NEW RECEIVE page.

Step2: Enter the unique PERSON ID of person.

Step3: Enter the Units of blood received by a person.

Step4: Enter the Hospital Name in which the person is being treated.

Step5: Click on SUBMIT button.

5.5.7 Stock Page:

If the passenger forgets password then he can recover it.

Step1: Go to HOME page

Step2: Press MOVIE button

Step3: select FORGOT PASSWORD

Step4: Enter the following fields

5.5.8 Donation History Page:

Specify the time interval so that you can get details from that time interval only.

Step1: Enter the time interval in after and before date.

Step2: Click on SEARCH button.

It displays all the donation history of that time interval.

If no donation history is present then it responds with “No record found on this time interval”.

5.5.9 Receive History Page:

Specify the time interval so that you can get details from that time interval only.

Step1: Enter the time interval in after and before date.

Step2: Click on SEARCH button.

It displays all the receive history of that time interval.

If no receive history is present then it responds with “No record found on this time interval”.

5.5.10 Add User Page:

Adding new user.

Step1: Enter the SUPER ADMIN PASSWORD.

Step2: Enter the new USERNAME.

Step3: Enter the PASSWORD.

Step4: CONFIRM PASSWORD.

Step5: Click on CREATE USER button.

New user is created.

5.5.11 Logout:

Once you are done with the work. You can logout using “LOGOUT” button.

CHAPTER 4

IMPLEMENTATION

4.1 Modules and their Roles

Login: Login for the new user.

```
<?php
Session_ start ();
include (‘connection. php’);
if(isset($_POST[‘submit’])) {
$username = stripcslashes($_ POST[‘user’]);
$username = my sqli_real_escape_string($con, $username);
$password = stripcslashes($_POST[‘pass’]);
$password = mysqli_real_escape_string($con, $password);
if(username == ‘’ or $password == ‘’){
$_SESSION[“login_error”] = ‘username or password cannot be empty’;
header (‘Location: index. php’);
}
else {
$sql = “select” * from user where username = ‘$username’ and password = ‘$password’;
$result = $con->query ($sql);
if($result->num_ rows >0) {
$row = $result ->fetch_ assoc();
$_ SESSION[“login”] = 1;
$_SESSION[“username”] = $row[“username”];
header (‘Location: Home.php’);


}
else {
$_SESSION[“login”] = 0;
$_SESSION[“login_error”] = ‘invalid login credentials’;
header (‘Location: index.php’);
```

```
        }
       }
}
else
   header (‘Location: index.php’);
?>
```

Add person:

```
<?php
session_ start ();
$_SESSION[“tab”] = “Add Person”;
if($_SESSION[“login”]! =1)
  echo ‘<h2 txtcolor = “red”>Authentication Error!!!</h2>’;
else {
  include_ once(‘header.php’);
  $name = $_ POST[‘name’];
  $phone = $_POST[‘phone’];
  $gender = $_POST[‘gender’];
  $dob = $_POST[‘dob’];
  $blood_ group = $_POST[‘blood_ group’];
  $address=$_POST[‘address’];
  $med_ issues = $_POST[‘med_ issues’];
 $sql = “insert into Person (p_name,p_phone,p_dob,p_address,p_gender,p_blood_group,p_med_isssues) values (‘$name’, ’$phone’, ‘$dob’, ’$address’, ’$gender’)
if ($con->query($sql) == TRUE){
   $sql = “select p_ id from Person where P_ name =’$name’ and p_ phone =’$phone’ and p_ gender =’$gender’ and p_ dob = ‘$dob’ and p_ blood_ group = ‘$blood_ group’;
$result = mysqli_ query($con, $sql);
$row = mysqli_ fetch_ array($result);
$count = mysqli_ num_ rows($result);
$pid = $row [‘p_ id’];
}
else
   echo “Error:” .  $sql . “<cbr>” . $con->error;
  ###########contents of div goes here#######
if($count == 1){
        echo'<h2>' .$name .'</h2><br><br>';
        echo 'Your registration is succesful..<br><br>';
        echo'Personal Id : '.$pid .'<br><br>';
        echo'Name : '.$name .'<br><br>';
        echo 'Phone Number: ' .$phone .'<br><br>';
        echo 'DOB : ' .$dob .'<br><br>';
        echo 'Blood Group : ' .$blood_group .'<br><br>';
        echo 'Gender : ';
        if ($gender === 'm')
            echo 'Male<br><br>';
        if ($gender === 'f')
            echo 'Female<br><br>';
        if ($gender === 'o')
            echo 'Others<br><br>';
        echo 'Address : ' .$address .'<br><br>';
        echo 'Medical Issues : ';
        if ($med_issues === "")
            echo 'None<br><br>';
        echo '<div style ="color:red;">NB:Please keep your Personal Id for future reference!!!';
    }
        ##################################################
    include_once('footer.php');
   }
  ?>
```

Add user:

```
        <?php
session_start();
$_SESSION["tab"] = "Add User";
if($_SESSION["login"] != 1)
  echo '<h2 txtcolor="red">Authentication Error!!!</h2>';
else{
  include_once('header.php');
    ###########contents of div goes here###########
  echo '
  <form name="addUser" action = "addUser.php"  method = "POST">
  <h2>Add User</h2>
  <br>
  <p>
mb4;
```

Add Receiver:

```
<?php
session_start();
$_SESSION["tab"] = "Add Person";
if($_SESSION["login"] != 1)
    echo '<h2>Access denied!!!</h2>';
else{
    include_once('header.php');
?>

<form name="addPerson" action="Add Receiver.php" method="POST">
    <h2>New Registration</h2>
    <br>

    <?php
    if(isset($_SESSION["entry_error"])){
```

```
    echo "<p class='error'>" .$_SESSION["entry_error"]. "</p>";
    unset($_SESSION["entry_error"]);
}
?>

<p>
<label>Name: </label>
<br>
<input type="text" name="name" class="input" />
</p>

<p>
<label>Phone Number: </label>
<br>
<input type="number" name="phone" maxlength="10" class="input" required />
</p>

<p>
<label>Gender:</label><br>
<input type="radio" id="male" name="gender" value="m" required>
<label for="male">Male</label><br>

<input type="radio" id="female" name="gender" value="f" required>
<label for="female">Female</label><br>

<input type="radio" id="other" name="gender" value="o" required>
<label for="other">Other</label>
</p>

<p>
<label>Date of birth: </label>
```

```
<br>
<input type="date" name="dob" class="input" required/>
</p>

<p>
<label>Blood Group:</label>
<br>
<select name="blood_group" class="input" required>
  <option value="A+">A+</option>
  <option value="A-">A-</option>
  <option value="B+">B+</option>
  <option value="B-">B-</option>
  <option value="O+">O+</option>
  <option value="O-">O-</option>
  <option value="AB+">AB+</option>
  <option value="AB-">AB-</option>
</select>
</p>

<p>
<label>Address:</label><br>
<textarea rows="5" cols="30" name="address" class="input area" required></textarea>
</p>

<p>
<label>Medical Issues (if any):</label>
<br>
<textarea rows="5" cols="30" name="med_issues" class="input area"></textarea>
</p>

<h2>Blood Receiver</h2>
```

```
    <p>
    <label>Receiver:</label><br>



    <input type="radio" id="receiver" name="donation_receiver" value="R" required>
    <label for="receiver">R</label>
    </p>

    <p>
    <label>Units of Blood:</label><br>
    <input type="number" name="units" class="input" required max="350">
    </p>

    <p>
    <button class="btn">Register</button>
    </p>
</form>

<?php
    include_once('footer.php');
}
?>
```

Check stock:

```
<?php
session_start();
$_SESSION["tab"] = "Check Stock";

if($_SESSION["login"] != 1)
        echo '<h2 txtcolor="red">Authentication Error!!!</h2>';
else{
        include_once('header.php');
```

```
                ###########contents of div goes here###########
        include_once('checkStock.php');

                ##################################################
        include_once('footer.php');
}
?>
```

New Donation:

```
<?php
session_start();
$_SESSION["tab"] = "New Donation";

if($_SESSION["login"] != 1)
        echo '<h2 txtcolor="red">Authentication Error!!!</h2>';
else{
        include_once('header.php');

                ###########contents of div goes here###########
        echo '
        <form name="newDonation" action = "newDonation.php"  method = "POST">
        <h2>New Donation</h2>
        <br>

        <p>
        <label>Personal Id:</label>
        <br>
        <input type = "Number" name  = "pid" class="input" required/>
        </p>

        <p>
```

```
	<label>Units of blood donated:</label>
	<br>
	<input type = "Number" maxlength="1" name  = "units" class="input" required/>
	</p>


	<p>
	<button class="btn">Submit</button>
	</p>
	</form>';

		####################################################
	include_once('footer.php');
}
?>
```

New Receive:

```
<?php
session_start();
$_SESSION["tab"] = "New Receive";

if($_SESSION["login"] != 1)
	echo '<h2 txtcolor="red">Authentication Error!!!</h2>';
else{
	include_once('header.php');

		###########contents of div goes here###########
	echo '<form name="addPerson" action = "newReceive.php"  method = "POST">
	<h2>New Receive</h2>
	<br>

	<p>
	<label>Personal Id:</label>
```

```
        <br>
        <input type = "Number" name  = "pid" class="input" required/>
        </p>

        <p>
        <label>Units of blood Received:</label>
        <br>
        <input type = "Number" maxlength="1" name  = "units" class="input" required/>
        </p>

        <p>
        <label>Admitted Hospital:</label>
        <br>
        <input type = "text" name  = "hospital" class="input" required/>
        </p>

        <p>
        <button class="btn">Submit</button>
        </p>




        </form>';

                ####################################################
        include_once('footer.php');
}
?>
```

SQL Commands

```
CREATE TABLE person (
  p_id int(10) NOT NULL,
  p_name varchar(25) NOT NULL,
  p_phone char(10) NOT NULL,
  p_dob date NOT NULL,
  p_address varchar(100) DEFAULT NULL,
  p_gender char(1) NOT NULL,
  p_blood_group varchar(3) NOT NULL,
  p_med_issues varchar(100) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE receive (
  p_id int(10) NOT NULL,
  r_date date NOT NULL,
  r_time time NOT NULL,
  r_quantity int(1) NOT NULL,
  r_hospital varchar(50) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE stock (
  s_blood_group varchar(3) NOT NULL,
  s_quantity int(5) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
INSERT INTO stock (s_blood_group, s_quantity) VALUES
('A+', 0),
('A-', 0),
('AB+', 0),
('AB-', 0),
('B+', 0),
('B-', 0),
('O+', 0),
('O-', 0);
CREATE TABLE user (
  username varchar(10) NOT NULL,
  password varchar(16) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

INSERT INTO user (username, password) VALUES
('SuperAdmin', '12345678'),
('test_user', 'qwertyuiop');

ALTER TABLE donation
  ADD PRIMARY KEY (p_id,d_date,d_time);

ALTER TABLE person
  ADD PRIMARY KEY (p_id);

ALTER TABLE receive
  ADD PRIMARY KEY (p_id,r_date,r_time);

ALTER TABLE stock
  ADD PRIMARY KEY (s_blood_group);

ALTER TABLE user
  ADD PRIMARY KEY (username);

ALTER TABLE person
  MODIFY p_id int(10) NOT NULL AUTO_INCREMENT;

ALTER TABLE donation
  ADD CONSTRAINT Donation_ibfk_1 FOREIGN KEY (p_id) REFERENCES person (p_id);
```

```
ALTER TABLE receive
  ADD CONSTRAINT Receive_ibfk_1 FOREIGN KEY (p_id) REFERENCES person (p_id);
COMMIT;
```

RESULT

The resulting system is able to:

- The person can fix their donation schedule using online reservation for donation of blood.
- The person can search for availability of required blood in the local blood bank in the case of emergency.
- The blood bank to store the details of the blood donated by person, like RBC, WBC, platelet count etc.

CHAPTER 5

TESTING

5.1 Software Testing

Software testing is a crucial process employed to assess the correctness, completeness, security, and overall quality of developed software. It involves the systematic execution of a program with the aim of identifying errors or defects. It's essential to differentiate between faults (issues in the code) and failures (issues observed by users). Through software testing, objective and independent insights about the software's quality and the risk of potential failures can be obtained, offering valuable information to users or sponsors. Testing can commence as soon as executable software, even if partially complete, is available, although most testing typically occurs after system requirements have been defined and implemented in testable programs.

5.2 Module Testing and Integration

Module testing focuses on testing individual components such as subprograms, subroutines, classes, or procedures within a program. Rather than testing the entire software program at once, module testing involves testing the smaller building blocks of the program. This approach is primarily white box-oriented, aiming not only to demonstrate proper functioning but also to identify errors within each module. Module testing facilitates the implementation of parallelism in the testing process, allowing multiple modules to be tested simultaneously.

Furthermore, integration testing ensures that the final integrated system has been thoroughly tested for various scenarios, including duplicate entries and type mismatches, among others. This phase ensures that the integrated software functions seamlessly and meets the specified requirements.

5.3 Limitations

- **Data accuracy**: Inaccurate or outdated information may lead to mismanagement of blood inventory or errors in matching donors with recipients.
- **Limited Accessibility:** If the system is not web-based or mobile-friendly, it may only be accessible from certain locations or devices, limiting its usability and efficiency.

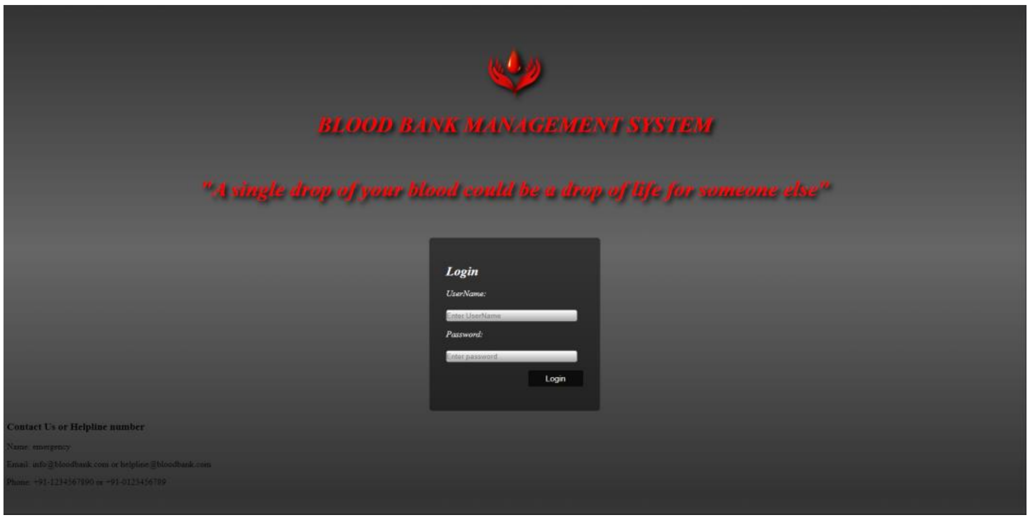
Temperature Monitoring: Blood products must be stored at specific temperatures to maintain their integrity. Ensuring proper temperature monitoring and control throughout the storage and transportation process is essential to prevent spoilage and maintain product quality

- **Cost constraints:** Implementing and maintaining a blood bank management system can be expensive, potentially limiting access to smaller blood banks or organizations with limited budgets.
- **Integration issues**: Difficulty in integrating with other healthcare systems may result in disjointed workflows and inefficiencies.
- **Expiration Management:** Blood products have limited shelf lives, and tracking their expiration dates accurately is crucial to prevent wastage. Managing the rotation of inventory to minimize expired units requires precise tracking and forecasting capabilities.

Chapter6

SNAPSHOTS

This chapter consists of working screenshots of the project.


BLOOD BANK MANAGEMENT SYSTEM
"A single drop of your blood could be a drop of life for someone else"
Login
UserName:
Enter UserName
Password:
Enter password
Login

Contact Us or Helpline number

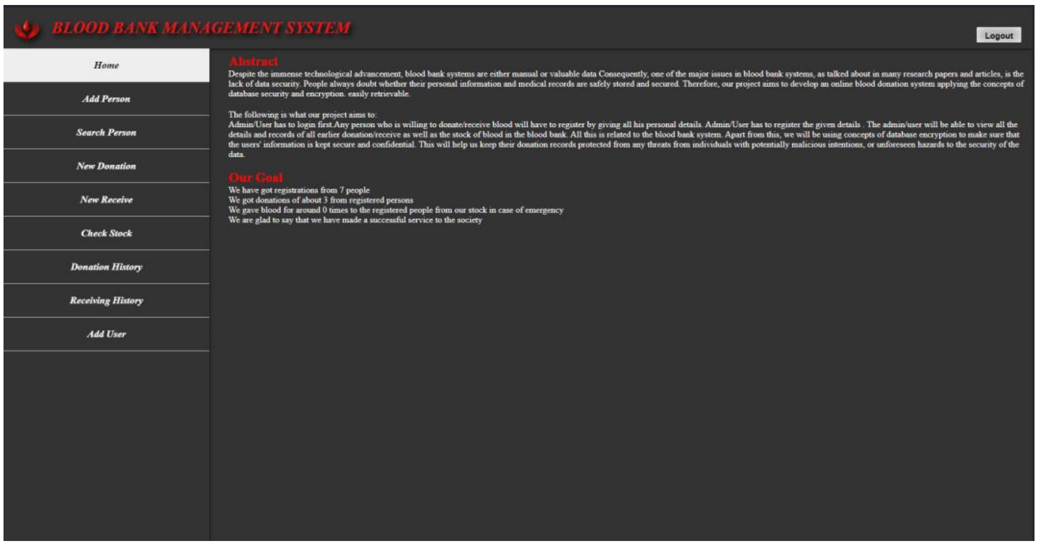
Name: emergency

Email: info@bloodbank.com or helpline@bloodbank.com

Phone: +91-1234567890 or +91-0123456789

Fig 6.1: Login Page

In the above fig 6.1 shows the login page, if the user wants to login, they must enter valid/registered username and password and click on login button.



Abstract

Despite the immense technological advancement, blood bank systems are either manual or valuable data Consequently, one of the major issues in blood bank systems, as talked about in many research papers and articles, is the lack of data security. People always doubt whether their personal information and medical records are safely stored and secured. Therefore, our project aims to develop an online blood donation system applying the concepts of database security and encryption. easily retrievable.

The following is what our project aims to:
Admin/User has to login first.Any person who is willing to donate/receive blood will have to register by giving all his personal details. Admin/User has to register the given details . The admin/user will be able to view all the details and records of all earlier donation/receive as well as the stock of blood in the blood bank. All this is related to the blood bank system. Apart from this, we will be using concepts of database encryption to make sure that the users' information is kept secure and confidential. This will help us keep their donation records protected from any threats from individuals with potentially malicious intentions, or unforeseen hazards to the security of the data.

Our Goal

We have got registrations from 7 people
We got donations of about 3 from registered persons
We gave blood for around 0 times to the registered people from our stock in case of emergency
We are glad to say that we have made a successful service to the society

Fig 6.2: Home page

In the above fig 6.2 shows the information about the blood bank management system.

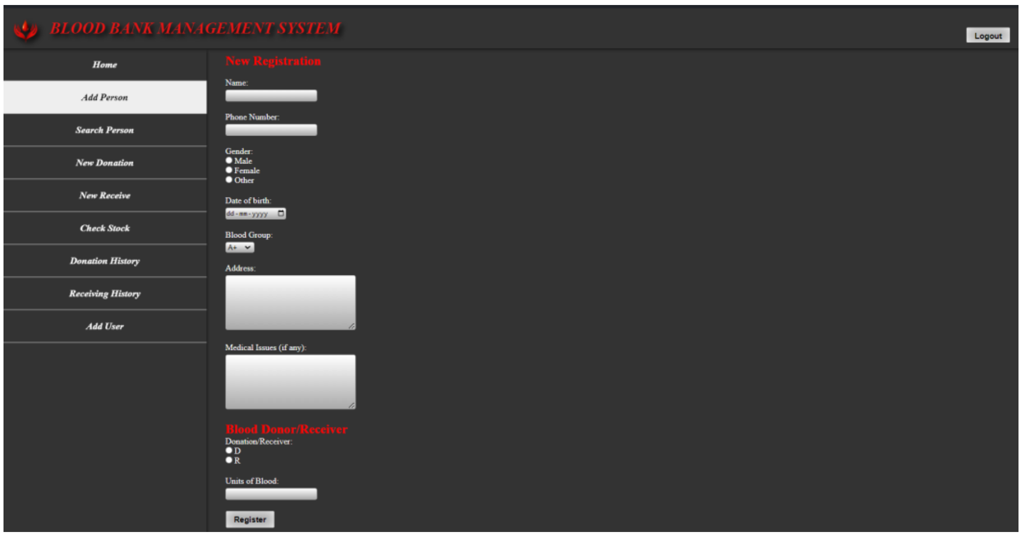
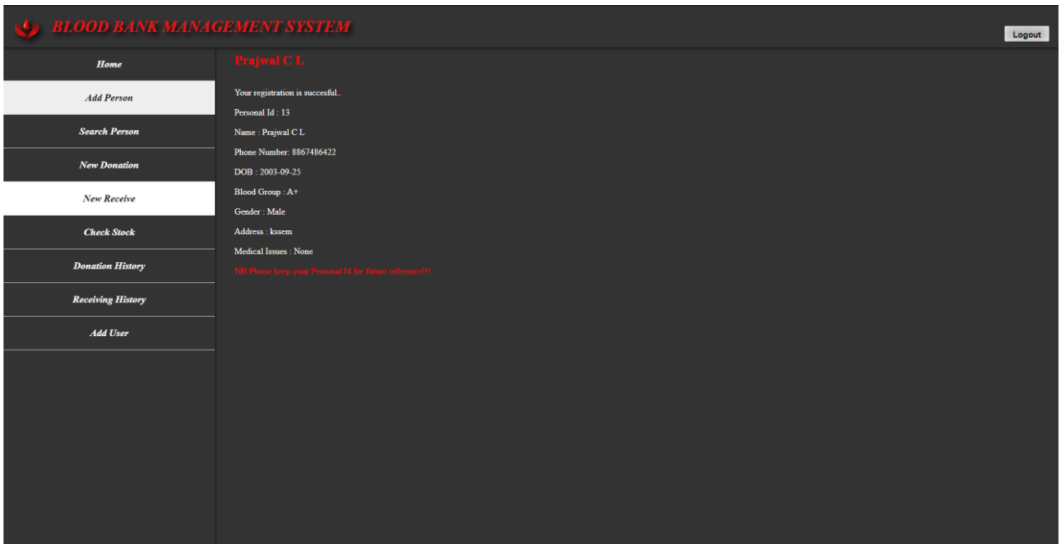


Fig 6.3: Add person page

In the above fig 6.3 shows the Add person page, here details of the donor/receiver is registered.



Prajwal C L

Your registration is succesful..

Personal Id : 13

Name : Prajwal C L

Phone Number: 8867486422

DOB : 2003-09-25

Blood Group : A+

Gender : Male

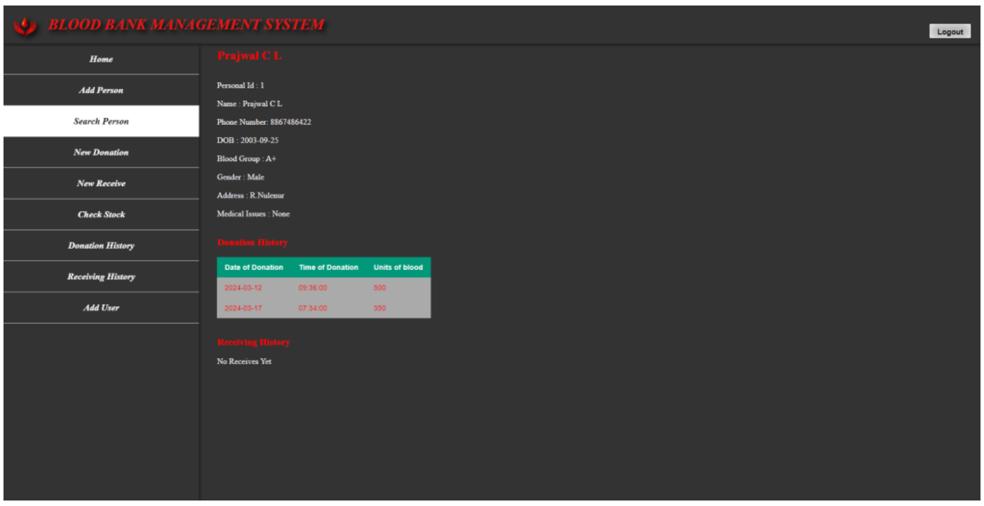
Address : kssem

Medical Issues : None

NB:Please keep your Personal Id for future reference!!!

Fig 6.3.1: Add person page

In the above fig 6.3.1 shows once entry of person details is done.



Prajwal C L

Personal Id : 1
Name : Prajwal C L
Phone Number: 8867486422
DOB : 2003-09-25
Blood Group : A+
Gender : Male
Address : R.Nulenur
Medical Issues : None

Donation History

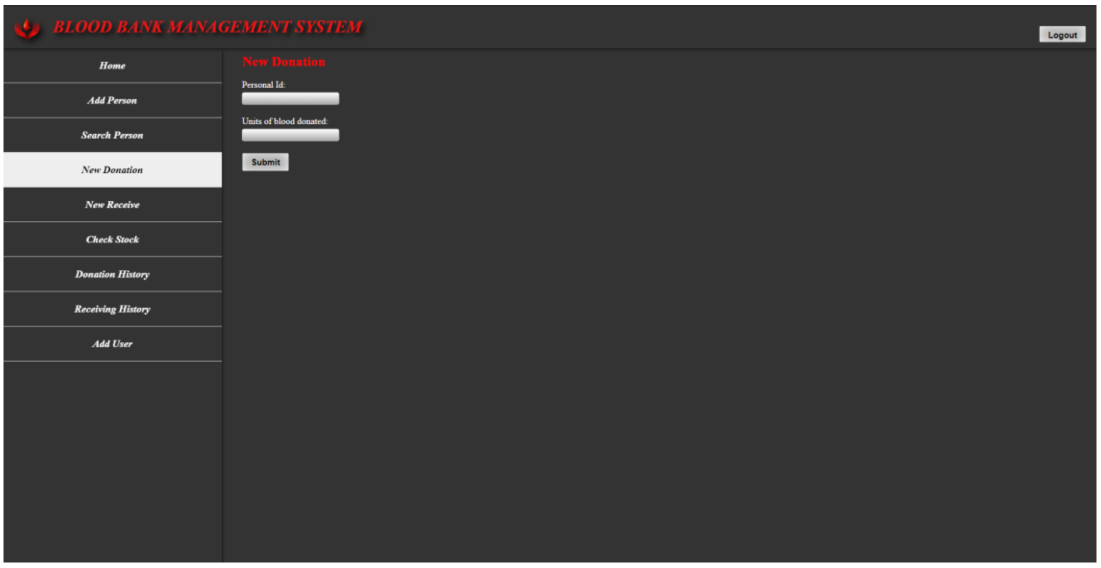
Date of Donation	Time of Donation	Units of blood
2024-03-12	09:36:00	500
2024-03-17	07:34:00	350

Receiving History

No Receives Yet

Fig 6.4: Search person page

In the fig 6.4 shows the result for search we done. If we enter the person id it will return with all the personal record present in the database.



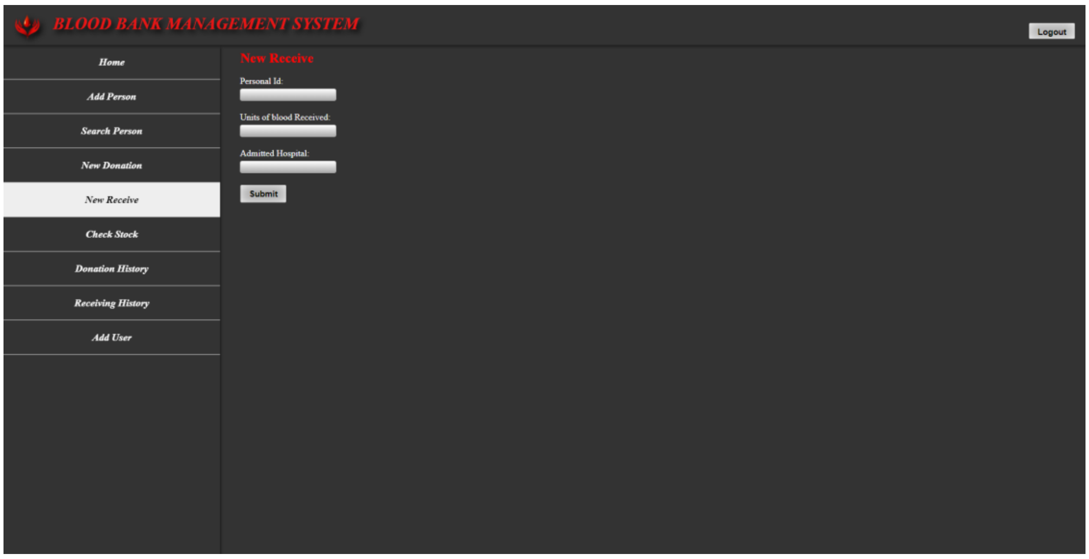
New Donation

Personal Id:

Units of blood donated:

Fig 6.5: Donation page

In the above fig 6.5 shows the New donation page, here person id and units of blood donated is recorded and submit button is clicked. On successful completion of entering it displays “Your donation is successful”.



New Receive

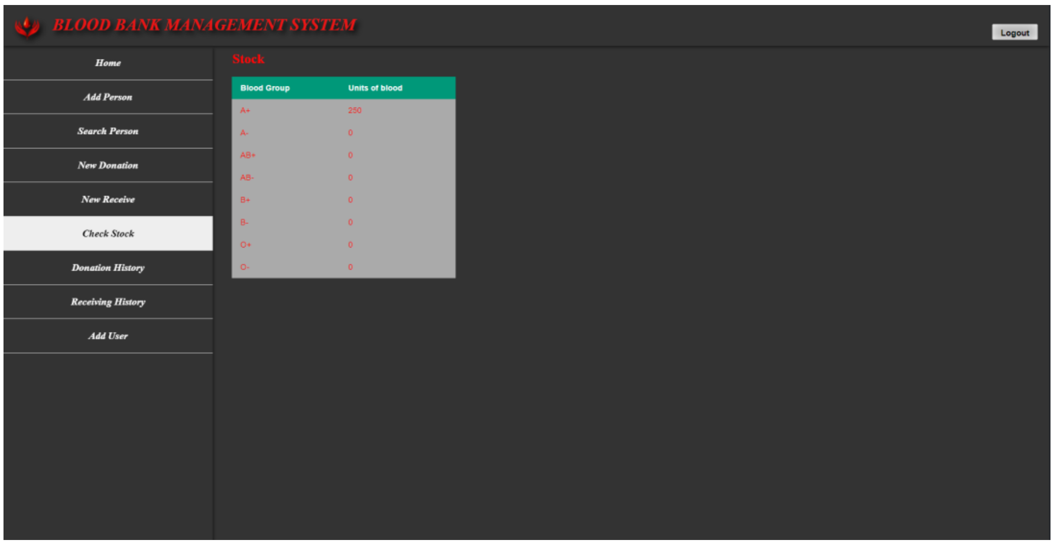
Personal Id:

Units of blood Received:

Admitted Hospital:

Fig 6.6: Receive page

In the above fig 6.6 shows the New Receive page, here person id, units of blood donated and hospital details is recorded and submit button is clicked. On successful completion of entering it displays "Your receiving is successful".

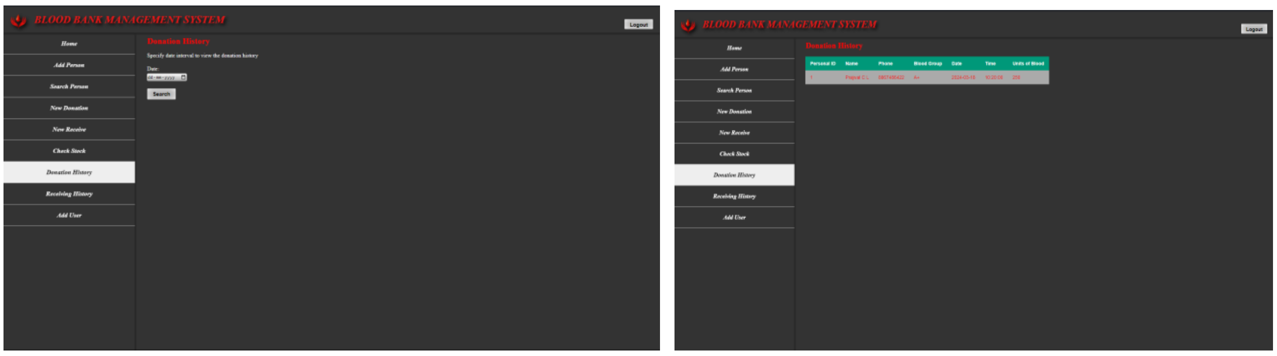


Stock

Blood Group	Units of blood
A+	250
A-	0
AB+	0
AB-	0
B+	0
B-	0
O+	0
O-	0

Fig 6.7: Check Stock page

In the above fig 6.7 shows the stock details i.e units of blood present in blood bank of each blood group.

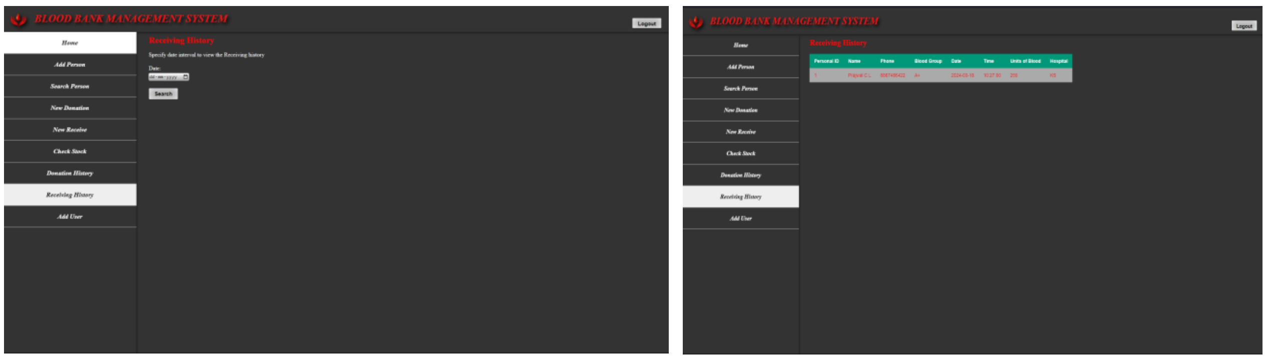


Donation History

Personal ID	Name	Phone	Blood Group	Date	Time	Units of Blood
1	Prajwal C L	8867486422	A+	2024-03-18	10:20:08	250

Fig 6.8: Donation History page

In the above fig 6.8 shows the Donation history page, the user has to give the time interval from when to when he wants to see the donation history details. Upon clicking submit he shows the donation history. If there is donation took place at that time interval is displays “No record found in the specified time interval”.



Receiving History

Personal ID	Name	Phone	Blood Group	Date	Time	Units of Blood	Hospital
1	Prajwal C L	8867486422	A+	2024-03-18	10:27:00	250	KS

Fig 6.9: Receive History page

In the above fig 6.9 shows the Receiving history page, the user has to give the time interval from when to when he wants to see the receive history details. Upon clicking submit he shows the receive history. If there is receiving of blood took place at that time interval is displays

“No record found in the specified time interval”.

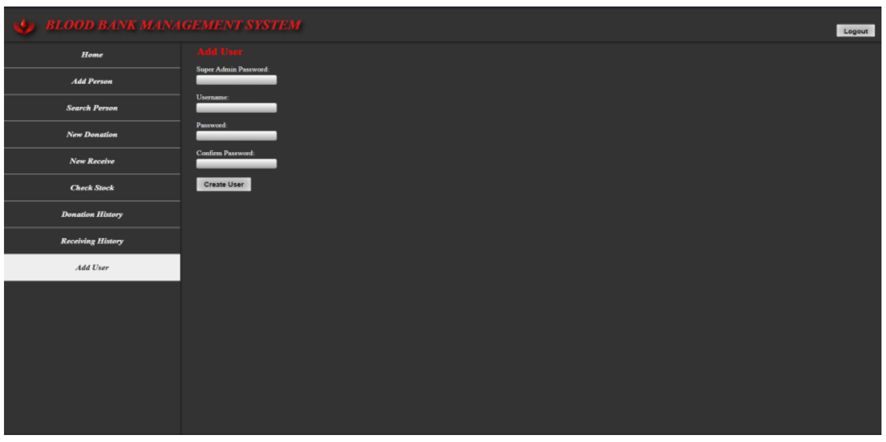


Fig 6.10: Add user page

In the above fig 6.10 shows the add user page, here it asks for super admin password. Next you have to enter the new username, password and confirm the password. Click on create user button. If the super admin password is wrong or password and confirm password don’t match. It will pop invalid password. On successfully creating user it will display “New user created successfully”.

CHAPTER 7

CONCLUSION

The main purpose of our blood management system is to provide blood bank with easier way to store and retrieve data and keep record of the availability of blood in blood bank.

After inserting the data to database, staff need not register of the same person again. They can simply search for recorded data and retrieve them for future blood donation or receiving purpose of that person.

CHAPTER 8

FUTURE ENHANCEMENTS

Future Upgrades to this project will implement:

In the nutshell, it can be summarized that the future scope of the project circles around maintaining information regarding:

- The person can fix their donation schedule using online reservation for donation of blood.
- The person can search for availability of required blood in the local blood bank in the case of emergency.
- The blood bank to store the details of the blood donated by person, like RBC, WBC, platelet count etc.

The above mentioned points are the enhancements which can be done to increase the applicability and usage of this project.

REFERENCES

- Silberschatz Korth and Sudharshan, Database System Concepts, 6th Edition, McGraw Hill, 2013.
- Fundamentals of Database Systems, Ramez Elmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.
- Database management systems, Ramakrishna, and Gehrke, 3rd Edition, 2014, McGrawHill, 2013.
- https://www.w3schools.com