Mini project report on

Blood Bank Management System

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering

UE21CS351A – DBMS Project

Submitted by:

NAGAVENI L G	PES2UG21CS315
NAVYA H U	PES2UG21CS325

Under the guidance of

Prof. Smrithi Surendran

Assistant Professor

Designation

PES University

AUG - DEC 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)

Electronic City, Hosur Road, Bengaluru – 560 100, Karnataka, India

CERTIFICATE

This is to certify that the mini project entitled

Blood Bank Management System

is a bonafide work carried out by

NAGAVENI L G	PES2UG21CS315
NAVYA H U	PES2UG21CS325

In partial fulfilment for the completion of fifth semester DBMS Project (UE20CSS351A) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period AUG. 2023 – DEC. 2023. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The project has been approved as it satisfies the 5^{th} semester academic requirements in respect of project work.

Signature

Prof. Smrithi Surendran

Assistant Professor

DECLARATION

We hereby declare that the DBMS Project entitled **Blood Bank Management System** has been carried out by us under the guidance of **Prof. Smrithi Surendran, Assistant Professor** and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester AUG – DEC 2023.

NAGAVENI L G	PES2UG21CS325	<SIGNATURE>
NAVYA H U	PES2UG21CS325	<SIGNATURE>

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Smrithi Surendran, Department of Computer Science and Engineering, PES University, for her continuous guidance, assistance, and encouragement throughout the development of this UE21CS351 - DBMS Project. I take this opportunity to thank Dr. Sandesh B J, C, Professor,ChairPerson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this DBMS Project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

The Blood Bank Management System (BBMS) is a comprehensive software solution designed to enhance and streamline the operations of blood banks within healthcare institutions, hospitals, and blood donation centres. The project's primary objective is to automate processes, centralize data, ensure secure access, and optimize resource allocation for the efficient management of blood-related activities. The system encompasses key components such as Login, Person, Donor, Receiver, and Stock, collectively providing a secure, user-friendly platform for managing blood bank operations.

The project addresses the limitations of existing manual systems, which often lead to time-consuming data retrieval and pose concerns about data security. The proposed system introduces a secure login mechanism for authorized personnel, allowing them to access and manage a centralized database. Personal details of donors and recipients are meticulously recorded and stored, enabling administrators to search and retrieve donation and receiving histories. The system also offers a comprehensive overview of blood stock, including real-time inventory monitoring and availability checks.

To address data security concerns, the project incorporates database encryption concepts. Admin and user roles are defined with varying levels of access and permissions to ensure confidentiality. This additional layer of security aims to protect personal information, donation records, and other sensitive data from potential threats and unforeseen hazards.

The major functionalities of the project include secure authentication, user role management, donor and recipient profile creation, donor registration, donation record keeping, blood receipt documentation, matching and verification processes, real-time inventory monitoring. These functionalities collectively contribute to a robust and efficient blood bank management system that improves patient care, simplifies tasks, and ensures the secure and confidential handling of sensitive data

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1.INTRODUCTION

The Blood Bank Management System (BBMS) project emerges as a pivotal response to the inefficiencies inherent in existing blood bank systems within healthcare institutions, hospitals, and blood donation centres. Current systems often grapple with manual data entry and retrieval, posing challenges in terms of time consumption and data security. In contrast, BBMS strives to revolutionize blood bank operations by automating processes, centralizing data, and ensuring secure access.

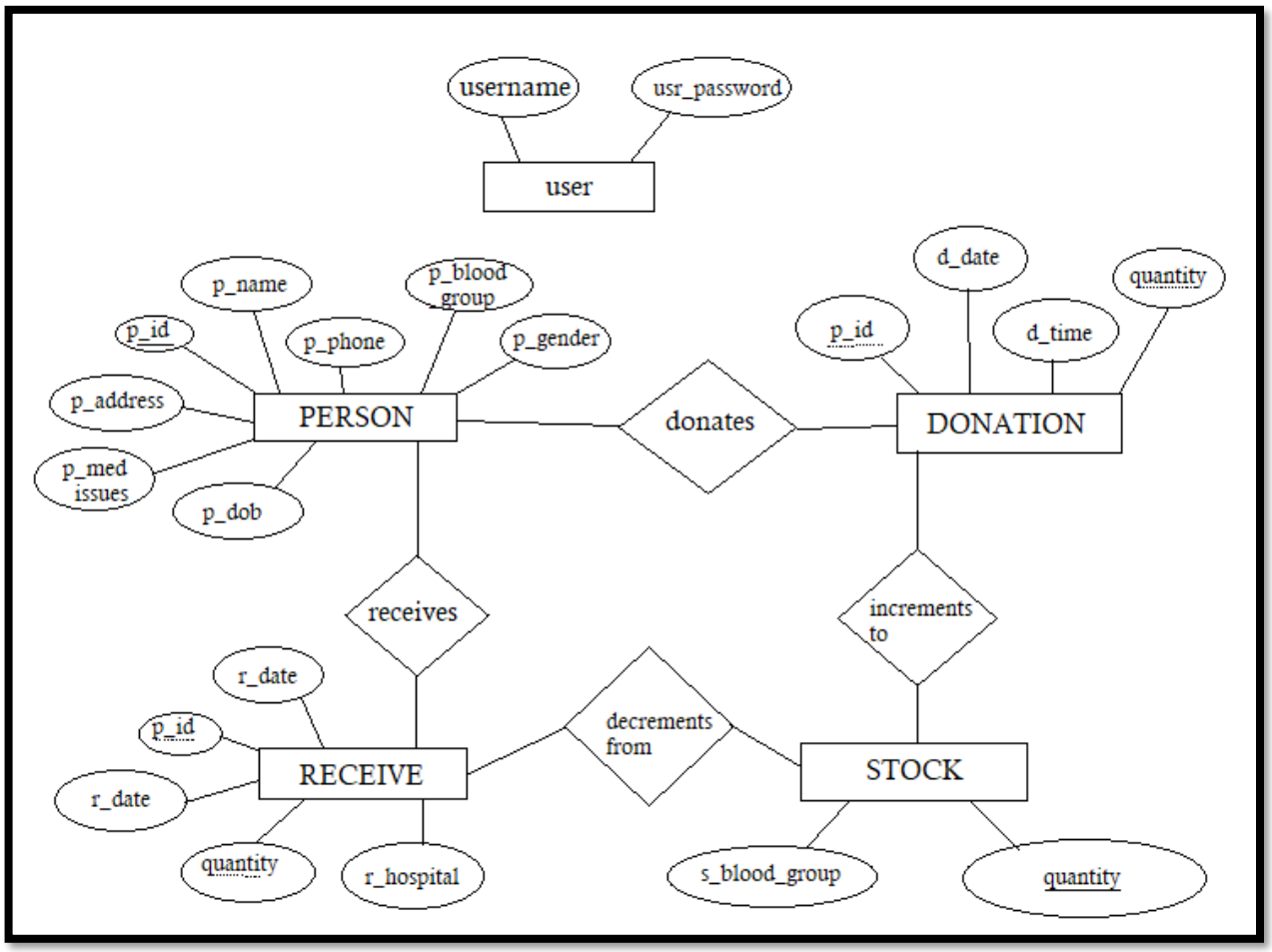
In the existing landscape, despite technological advancements, blood bank systems rely on manual data storage, leading to cumbersome retrieval and heightened concerns about data security. The proposed BBMS project presents a holistic solution, introducing a secure login mechanism, comprehensive donor and recipient profiles, and real-time inventory monitoring. By addressing these critical facets, BBMS not only streamlines operations but also enhances the safety and efficiency of managing blood donations and distributions.

The project's scope encompasses a secure authentication process, user role management, and the implementation of database encryption for heightened security. BBMS introduces functionalities such as donor registration, detailed donation records, blood receipt documentation. Through a user-friendly interface, BBMS aims to improve patient care by offering a centralized, secure, and efficient platform for blood bank activities. In essence, BBMS signifies a leap forward, promising to bridge the gaps in existing systems and pave the way for a more effective and secure management of blood-related data and activities

2. PROBLEM DEFINATION

The current challenge in blood bank systems lies in the absence of a fully functional, efficient platform that allows users to seamlessly log in, donate blood, and maintain detailed records. The existing manual processes result in inefficiencies and pose obstacles to quick and secure data management. The need is for a comprehensive solution that not only enables secure user authentication but also streamlines the donation process, ensuring that detailed information is efficiently recorded and managed. The objective is to create a system that facilitates user interaction, promotes efficient blood donation, and maintains comprehensive records in a secure and user-friendly manner. The Blood Bank Management System (BBMS) project is initiated to address these specific needs and provide a robust platform for managing blood-related activities within healthcare institutions.

3. ER MODEL


username
usr_password
user
p_name
p_blood group
p_id
p_phone
p_gender
p_address
PERSON
p_med issues
p_dob
donates
d_date
quantity
p_id
d_time
DONATION
receives
increments to
r_date
p_id
r_date
RECEIVE
quantity
r_hospital
decrements from
STOCK
s_blood_group
quantity

4. ER TO RELATIONAL MAPPING

4.1 STEPS OF ALGORITHM FOR CHOSEN PROBLEM

1. **Define Database Schema:**
    - Entities: User, Person, Receive, Donation, Stock
    - Relationships: One-to-Many and Many-to-One relationships among entities
    - Attributes: Various attributes for each entity

2. **Create Database Tables:**
    - SQL statements to create tables for User, Person, Receive, Donation, Stock
    - Set primary keys, foreign keys, and constraints.

3. **Implement User Authentication:**
    - Algorithm Login():
        - Step 1: Input USERNAME in the given field.
        - Step 2: Input valid PASSWORD.
        - Step 3: Click on LOGIN button.

4. **Manage Donors and Receivers:**
    - Algorithm AddPerson():
        - Step 1: Open the ADD PERSON page.
        - Step 2: Auto-increment PERSON ID (unique for all).
        - Step 3: Input PERSON NAME.
        - Step 4: Input PHONE NUMBER.
        - Step 5: Input GENDER.
        - Step 6: Select DATE OF BIRTH.
        - Step 7: Select BLOOD GROUP.
        - Step 8: Input ADDRESS.
        - Step 9: Input MEDICAL ISSUES if any.

- Step 10: Click on REGISTER button.

- Algorithm SearchPerson():
    - Step 1: Open the SEARCH PERSON page.
    - Step 2: Input PERSON ID.
    - Step 3: Click on SUBMIT button.

5. **Blood Donation and Receipt:**
    - Algorithm Donation():
        - Step 1: Open the NEW DONATION page.
        - Step 2: Input unique PERSON ID of the person.
        - Step 3: Input Units of blood donated.
        - Step 4: Click on SUBMIT button.

    - Algorithm Receive():
        - Step 1: Open the NEW RECEIVE page.
        - Step 2: Input unique PERSON ID of the person.
        - Step 3: Input Units of blood received.
        - Step 4: Input Hospital Name.
        - Step 5: Click on SUBMIT button.

6. **Real-time Blood Stock Monitoring:**
    - Display current stock levels for different blood groups.

7. **Search and Retrieval:**
    - Enable users to search for donor or receiver details based on various parameters.

8. **Data Security Measures:**
    - Implement database encryption to ensure the confidentiality of sensitive information.
    - Enforce secure access controls based on user roles.

9. **Donation and Receive History:**

- Algorithm DonationHistory():
    - Step 1: Specify time interval (after and before date).
    - Step 2: Click on SEARCH button.
    - Display all donation history within the specified time interval.
    - If no donation history is present, respond with "No record found on this time interval."

- Algorithm ReceiveHistory():
    - Step 1: Specify time interval (after and before date).
    - Step 2: Click on SEARCH button.
    - Display all receive history within the specified time interval.
    - If no receive history is present, respond with "No record found on this time interval."

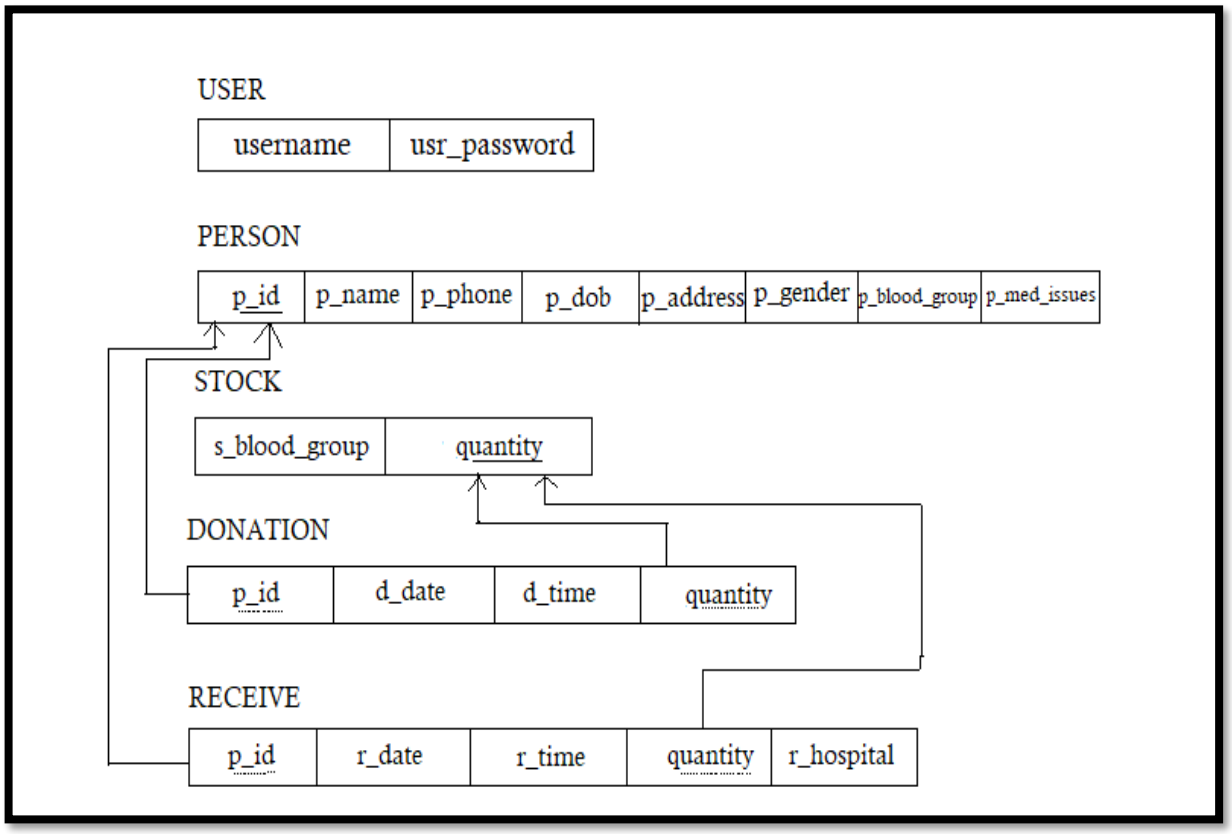
11. **Add User:**

- Algorithm AddUser():
    - Step 1: Enter SUPER ADMIN PASSWORD.
    - Step 2: Enter new USERNAME.
    - Step 3: Enter PASSWORD.
    - Step 4: Confirm PASSWORD.
    - Step 5: Click on CREATE USER button.
    - New user is created.

12. **Logout:**

- Algorithm Logout():
    - Once done with work, logout using "LOGOUT" button.

4.2 COMPLETE DIAGRAM OF RELATIONAL MAPPING



5.DDL STATEMENTS

5.1.Database Creation

Creating ‘blood_bank’ Database

```
14      -- Database: `blood_bank`
15      --
16  •   CREATE DATABASE IF NOT EXISTS `blood_bank` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
17  •   USE `blood_bank`;
18
19      -- --------------------------------------------------------
20
```

5.2. Tables Creation

5.2.1 Creation of ‘user’ Table

```
113     -- Table structure for table `user`
114     --
115
116 •   CREATE TABLE `user` (
117       `username` varchar(10) NOT NULL,
118       `password` varchar(16) NOT NULL
119     ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
120
```

```
mysql> desc user;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| username | varchar(10) | NO   | PRI | NULL    |       |
| password | varchar(16) | NO   |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

5.2.2 Creation of ‘person’ Table

```
52    --
53    -- Table structure for table `person`
54    --
55
56    CREATE TABLE `person` (
57      `p_id` int(10) NOT NULL,
58      `p_name` varchar(25) NOT NULL,
59      `p_phone` char(10) NOT NULL,
60      `p_dob` date NOT NULL,
61      `p_address` varchar(100) DEFAULT NULL,
62      `p_gender` char(1) NOT NULL,
63      `p_blood_group` varchar(3) NOT NULL,
64      `p_med_issues` varchar(100) DEFAULT NULL
65    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
66
```

```
mysql> desc person;
+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
+---------------+--------------+------+-----+---------+----------------+
| p_id          | int          | NO   | PRI | NULL    | auto_increment |
| p_name        | varchar(25)  | NO   |     | NULL    |                |
| p_phone       | char(10)     | NO   |     | NULL    |                |
| p_dob         | date         | NO   |     | NULL    |                |
| p_address     | varchar(100) | YES  |     | NULL    |                |
| p_gender      | char(1)      | NO   |     | NULL    |                |
| p_blood_group | varchar(3)   | NO   |     | NULL    |                |
| p_med_issues  | varchar(100) | YES  |     | NULL    |                |
| donation_ban  | date         | YES  |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
9 rows in set (0.01 sec)
```

5.2.3 Creation of ‘donation’ Table

```
21    --
22    -- Table structure for table `donation`
23    --
24
25    CREATE TABLE `donation` (
26      `p_id` int(10) NOT NULL,
27      `d_date` date NOT NULL,
28      `d_time` time NOT NULL,
29      `d_quantity` int(1) NOT NULL
30    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
31
32
```

```
mysql> desc donation;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| p_id       | int  | NO   | PRI | NULL    |       |
| d_date     | date | NO   | PRI | NULL    |       |
| d_time     | time | NO   | PRI | NULL    |       |
| d_quantity | int  | NO   |     | NULL    |       |
+------------+------+------+-----+---------+-------+
4 rows in set (0.02 sec)
```

5.2.4 Creation of 'receive' Table

```
73    --
74    -- Table structure for table `receive`
75    --
76
77 •  CREATE TABLE `receive` (
78      `p_id` int(10) NOT NULL,
79      `r_date` date NOT NULL,
80      `r_time` time NOT NULL,
81      `r_quantity` int(1) NOT NULL,
82      `r_hospital` varchar(50) NOT NULL
83    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
84
```

```
mysql> desc receive;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| p_id       | int         | NO   | PRI | NULL    |       |
| r_date     | date        | NO   | PRI | NULL    |       |
| r_time     | time        | NO   | PRI | NULL    |       |
| r_quantity | int         | NO   |     | NULL    |       |
| r_hospital | varchar(50) | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
5 rows in set (0.01 sec)
```

5.2.5 Creation of 'stock' Table

```
88    -- Table structure for table `stock`
89    --
90
91    CREATE TABLE `stock` (
92      `s_blood_group` varchar(3) NOT NULL,
93      `s_quantity` int(5) NOT NULL DEFAULT 0
94    ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
95
```

```
mysql> desc stock;
+---------------+------------+------+-----+---------+-------+
| Field         | Type       | Null | Key | Default | Extra |
+---------------+------------+------+-----+---------+-------+
| s_blood_group | varchar(3) | NO   | PRI | NULL    |       |
| s_quantity    | int        | NO   |     | 0       |       |
+---------------+------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

6.DML STATEMENTS

6.1 Insertion of values into table ‘user’

```
122     -- Dumping data for table `user`
123     --
124
125 •   INSERT INTO `user` (`username`, `password`) VALUES
126     ('SuperAdmin', 'superadmin'),
127     ('test_user', 'testuser');
128
129     --
```

```
mysql>
mysql> select * from user;
+------------+------------+
| username   | password   |
+------------+------------+
| SuperAdmin | 12345678   |
| test_user  | qwertyuiop |
| testuser1  | testuser1  |
| testuser2  | testuser2  |
+------------+------------+
4 rows in set (0.05 sec)
```

6.2 Insertion of values into table ‘stock’

```
97      -- Dumping data for table `stock`
98      --
99
100 •   INSERT INTO `stock` (`s_blood_group`, `s_quantity`) VALUES
101     ('A+', 0),
102     ('A-', 0),
103     ('AB+', 0),
104     ('AB-', 0),
105     ('B+', 0),
106     ('B-', 0),
107     ('O+', 0),
108     ('O-', 0);
109
110     -- --------------------------------------------------------
111
```

```
mysql> select * from stock;
+---------------+------------+
| s_blood_group | s_quantity |
+---------------+------------+
| A-            |          0 |
| A+            |          2 |
| AB-           |          1 |
| AB+           |         -1 |
| B-            |          0 |
| B+            |          2 |
| O-            |          0 |
| O+            |          2 |
+---------------+------------+
8 rows in set (0.00 sec)
```

6.3 Insertion in procedure ‘addPersonProcedure’ through frontend

```
BEGIN
    -- Insert a new record
    INSERT INTO Person (p_name, p_phone, p_gender, p_dob, p_blood_group, p_address, p_med_issues)
    VALUES (p_name, p_phone, p_gender, p_dob, p_blood_group, p_address, p_med_issues);
```

```
mysql> select * from person;
+------+--------------+------------+------------+-----------------------------------------------------------+----------+--------------+-------------------
-+--------------+
| p_id | p_name       | p_phone    | p_dob      | p_address                                                 | p_gender | p_blood_group | p_med_issues
 | donation_ban |
+------+--------------+------------+------------+-----------------------------------------------------------+----------+--------------+-------------------
-+--------------+
|    1 | Nagaveni L G | 9380158819 | 2003-12-19 | Amaatra academy
Off Sarjapur road kasavanahalli Main Road | o        | A+           | NO                | 2023-11-14   |
|    2 | Navya        | 7676184299 | 2004-05-12 | Amaatra academy
Off Sarjapur road kasavanahalli Main Road | f        | A+           | Handicapt         | 2023-11-14   |
|    3 | Smriti Ajay  | 8762399878 | 1987-04-12 | Bengaluru                                                 | f        | A+           | NA
 | 2023-11-14   |
|    4 | Nishat E     | 9869565476 | 2002-03-17 | Amaatra academy
Off Sarjapur road kasavanahalli Main Road | f        | B+           | Mentally Ill      | 2023-11-14   |
|    5 | Abhay        | 9901696663 | 2005-09-12 | Bengaluru                                                 | m        | AB-          | Handicapt
 | 2023-11-14   |
|    6 | Abhay        | 9901696663 | 2005-09-12 | Bengaluru                                                 | m        | AB-          | Handicapt
 | NULL         |
|    7 | Mithun       | 9380158819 | 2010-07-04 | Tumakuru                                                  | m        | AB+          | NA
 | 2023-11-14   |
|    8 | Shravya N    | 8050360365 | 2004-04-10 | Amaatra academy
Off Sarjapur road kasavanahalli Main Road | f        | B+           | NA                | 2023-11-14   |
|    9 | Smriti Raj   | 9380158819 | 2004-05-15 | Amaatra academy
Off Sarjapur road kasavanahalli Main Road | f        | B+           | na                | 2023-11-14   |
|   10 | Navya H U    | 9380158819 | 2003-07-12 | Amaatra academy
Off Sarjapur road kasavanahalli Main Road | f        | A+           | mentally reatarded | 2023-11-16   |
|   11 | Nikita       | 6578872639 | 2003-11-27 | Amaatra academy
Off Sarjapur road kasavanahalli Main Road | f        | O+           | na                | 2023-11-16   |
|   12 | nikitha v    | 9380158819 | 2003-10-12 | Amaatra academy
Off Sarjapur road kasavanahalli Main Road | f        | AB-          | Na                | NULL         |
|   13 | Nagaveni L G | 9380158819 | 2003-12-19 | Amaatra academy
Off Sarjapur road kasavanahalli Main Road | f        | O+           | Na                | 2023-12-04   |
+------+--------------+------------+------------+-----------------------------------------------------------+----------+--------------+-------------------
-+--------------+
13 rows in set (0.02 sec)
```

7. QUERIES

7.1 SIMPLE QUERY USING AGRREGATE FUNCTION

```
$sql = "select count(p_id) from Person";
$result = mysqli_query($con, $sql);
$row = mysqli_fetch_array($result);
echo "We have got registrations from ".$row[0] ." people";
$sql = "select count(p_id) from Donation";
$result = mysqli_query($con, $sql);
$row = mysqli_fetch_array($result);
echo "<br>We got donations of about ".$row[0] ." from registered persons";
$sql = "select count(p_id) from Receive";
$result = mysqli_query($con, $sql);
$row = mysqli_fetch_array($result);
echo "<br>We gave blood for around ".$row[0] ." times to the registered people from our stock in case of emergency<br>";
echo "We are glad to say that we have made a successful service to the society</p>";
```

7.2 UPDATE OPERATION

Update in new receive tab to update ‘stock’ table

```
$sql_2 = "update  Stock SET s_quantity = s_quantity - '$units' where Stock.s_blood_group = (select p_blood_group FROM Person where p_id = '$pid')";
```

Update in new donation tab to update ‘stock’ table

```
$sql_2 = "UPDATE stock SET s_quantity = s_quantity + '$units' WHERE s_blood_group = (SELECT p_blood_group FROM person WHERE p_id = '$pid')";
```

7.3 CORRELATED QUERY

-> Uses a correlated subquery to find the maximum donation date for the person being inserted and updates the donation_ban accordingly.

```
-- Get the donation date for the newly inserted row
SET donation_date = NEW.d_date;

-- Get the person_id for the newly inserted row
SET person_id = NEW.p_id;

-- Update the person table using a nested query
UPDATE person
SET donation_ban = (
    SELECT DATE_ADD(MAX(d_date), INTERVAL 15 DAY)
    FROM donation
    WHERE p_id = person_id
)
WHERE p_id = person_id;
```

->The correlated query (UPDATE AnotherTable ... WHERE related_p_id = LAST_INSERT_ID();) updates another table (AnotherTable) based on the newly inserted p_id. Replace AnotherTable and related_p_id with your actual table and column names

```
BEGIN
    -- Insert a new record
    INSERT INTO Person (p_name, p_phone, p_gender, p_dob, p_blood_group, p_address,
    VALUES (p_name, p_phone, p_gender, p_dob, p_blood_group, p_address, p_med_issues

    -- Retrieve the newly inserted p_id
    SELECT LAST_INSERT_ID() AS p_id;

    -- Example correlated query: Update another table based on the newly inserted p_
    UPDATE AnotherTable
    SET some_column = CONCAT('Updated value for ', p_name)
    WHERE related_p_id = LAST_INSERT_ID();
END;
```

-> the subquery (SELECT COUNT(*) FROM Person at WHERE at.related_p_id = p.p_id) AS additional_count is a correlated subquery. It is correlated with the outer query by referencing the p.p_id from the outer query.

```
DELIMITER //
CREATE PROCEDURE GetReceiveHistory(IN p_sdate DATE, IN p_edate DATE)
BEGIN
    -- Use a correlated query to retrieve receive history with additional information
    SELECT
        p.p_id,
        p.p_name,
        p.p_phone,
        p.p_blood_group,
        r.r_date,
        r.r_time,
        r.r_quantity,
        (SELECT COUNT(*) FROM person at WHERE at.related_p_id = p.p_id) AS additional_count
    FROM Person p
    JOIN Receive r ON p.p_id = r.p_id
    WHERE r.r_date >= p_sdate AND r.r_date <= p_edate;
END;
//
DELIMITER ;
```

7.4 NESTED QUERY

```
DELIMITER //
CREATE PROCEDURE AddPersonProcedure(
    IN p_name VARCHAR(25),
    IN p_phone CHAR(10),
    IN p_gender CHAR(1),
    IN p_dob DATE,
    IN p_blood_group VARCHAR(3),
    IN p_address VARCHAR(100),
    IN p_med_issues VARCHAR(100)
)
BEGIN
    -- Insert a new record
    INSERT INTO Person (p_name, p_phone, p_gender, p_dob, p_blood_group, p_address, p_med_issues)
    VALUES (p_name, p_phone, p_gender, p_dob, p_blood_group, p_address, p_med_issues);

    -- Retrieve the newly inserted p_id and additional information
    SELECT
        LAST_INSERT_ID() AS p_id,
        p_name,
        p_phone,
        p_gender,
        p_dob,
        p_blood_group,
        p_address,
        p_med_issues
    FROM Person
    WHERE p_id = LAST_INSERT_ID();
END;
//
DELIMITER ;
```

```
DELIMITER //
CREATE PROCEDURE AddUserProcedure(
    IN p_super_pwd VARCHAR(16),
    IN p_usr_name VARCHAR(10),
    IN p_usr_pwd VARCHAR(16)
)
BEGIN
    DECLARE super_pwd_valid INT;
    DECLARE username_available INT;

    -- Check if the Super Admin password is valid
    SELECT COUNT(*) INTO super_pwd_valid FROM User WHERE username = 'SuperAdmin' AND password = p_super_pwd;

    IF super_pwd_valid = 1 THEN
        -- Check if the username is available
        SELECT COUNT(*) INTO username_available FROM User WHERE username = p_usr_name;

        IF username_available = 0 THEN
            -- Insert the new user record
            INSERT INTO User (username, password) VALUES (p_usr_name, p_usr_pwd);
        ELSE
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Username is not available.';
        END IF;
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Invalid Super Admin Password.';
    END IF;
END;
//
DELIMITER ;
```

8. STORED PROCEDURES, FUCNTIONS AND TRIGGERS

8.1 STORED PROCEDURES OR FUNCTIONS

```
DELIMITER //
CREATE PROCEDURE GetReceiveHistory(IN p_sdate DATE, IN p_edate DATE)
BEGIN
    -- Use a correlated query to retrieve receive history with additional information
    SELECT
        p.p_id,
        p.p_name,
        p.p_phone,
        p.p_blood_group,
        r.r_date,
        r.r_time,
        r.r_quantity,
        (SELECT COUNT(*) FROM person at WHERE at.related_p_id = p.p_id) AS additional_count
    FROM Person p
    JOIN Receive r ON p.p_id = r.p_id
    WHERE r.r_date >= p_sdate AND r.r_date <= p_edate;
END;
//
DELIMITER ;
```

```
DELIMITER //
CREATE PROCEDURE AddPersonProcedure(
    IN p_name VARCHAR(25),
    IN p_phone CHAR(10),
    IN p_gender CHAR(1),
    IN p_dob DATE,
    IN p_blood_group VARCHAR(3),
    IN p_address VARCHAR(100),
    IN p_med_issues VARCHAR(100)
)
BEGIN
    -- Insert a new record
    INSERT INTO Person (p_name, p_phone, p_gender, p_dob, p_blood_group, p_address, p_med_issues)
    VALUES (p_name, p_phone, p_gender, p_dob, p_blood_group, p_address, p_med_issues);

    -- Retrieve the newly inserted p_id and additional information
    SELECT
        LAST_INSERT_ID() AS p_id,
        p_name,
        p_phone,
        p_gender,
        p_dob,
        p_blood_group,
        p_address,
        p_med_issues
    FROM Person
    WHERE p_id = LAST_INSERT_ID();
END;
//
DELIMITER ;
```

```
DELIMITER //
CREATE PROCEDURE AddUserProcedure(
    IN p_super_pwd VARCHAR(16),
    IN p_usr_name VARCHAR(10),
    IN p_usr_pwd VARCHAR(16)
)
BEGIN
    DECLARE super_pwd_valid INT;
    DECLARE username_available INT;

    -- Check if the Super Admin password is valid
    SELECT COUNT(*) INTO super_pwd_valid FROM User WHERE username = 'SuperAdmin' AND password = p_super_pwd;

    IF super_pwd_valid = 1 THEN
        -- Check if the username is available
        SELECT COUNT(*) INTO username_available FROM User WHERE username = p_usr_name;

        IF username_available = 0 THEN
            -- Insert the new user record
            INSERT INTO User (username, password) VALUES (p_usr_name, p_usr_pwd);
        ELSE
            SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Username is not available.';
        END IF;
    ELSE
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Invalid Super Admin Password.';
    END IF;
END;
//
DELIMITER ;
```

```
DELIMITER //
CREATE FUNCTION GetDonationHistory(p_sdate DATE, p_edate DATE)
RETURNS TABLE (
    p_id INT,
    p_name VARCHAR(255),
    p_phone CHAR(10),
    p_blood_group VARCHAR(3),
    d_date DATE,
    d_time TIME,
    d_quantity INT
)
READS SQL DATA
BEGIN
    DECLARE done INT DEFAULT 0;
    DECLARE cur CURSOR FOR
        SELECT
            p.p_id,
            p.p_name,
            p.p_phone,
            p.p_blood_group,
            d.d_date,
            d.d_time,
            d.d_quantity
        FROM Person p
        JOIN Donation d ON p.p_id = d.p_id
        WHERE d.d_date >= p_sdate AND d.d_date <= p_edate;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO p_id, p_name, p_phone, p_blood_group, d_date, d_time, d_quantity;
        IF done = 1 THEN
            LEAVE read_loop;
        END IF;
        INSERT INTO result_table VALUES (p_id, p_name, p_phone, p_blood_group, d_date, d_time, d_quantity);
    END LOOP;

    CLOSE cur;
END;
//
DELIMITER ;
```

8.2 TRIGGERS

```
DELIMITER //
CREATE TRIGGER DonationBanTrigger
AFTER INSERT ON donation
FOR EACH ROW
BEGIN
    DECLARE donation_date DATE;
    SET donation_date = NEW.d_date;

    UPDATE person
    SET donation_ban = DATE_ADD(donation_date, INTERVAL 15 DAY)
    WHERE p_id = NEW.p_id;
END;
//
DELIMITER ;


-- ------------------------------------------------------
```

8.FRONT END DEVELOPEMNT

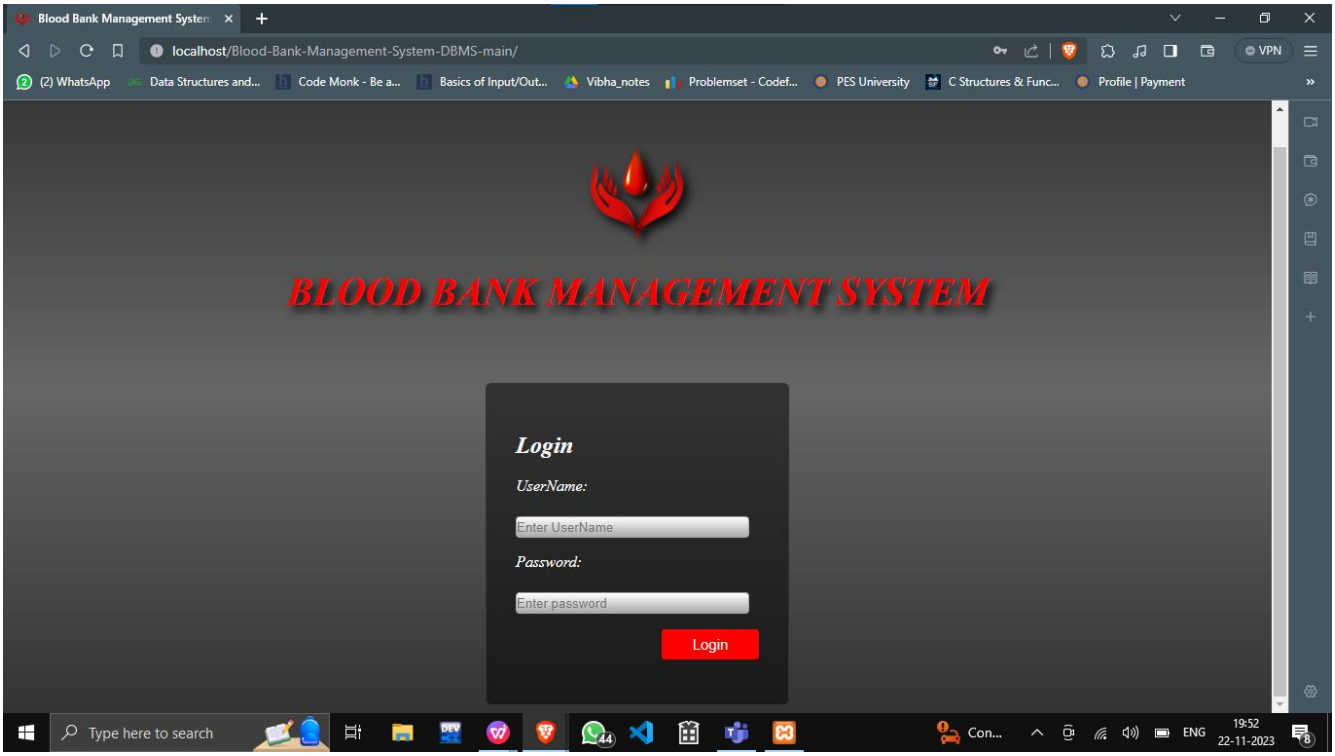


Fig 8.1: Login Page

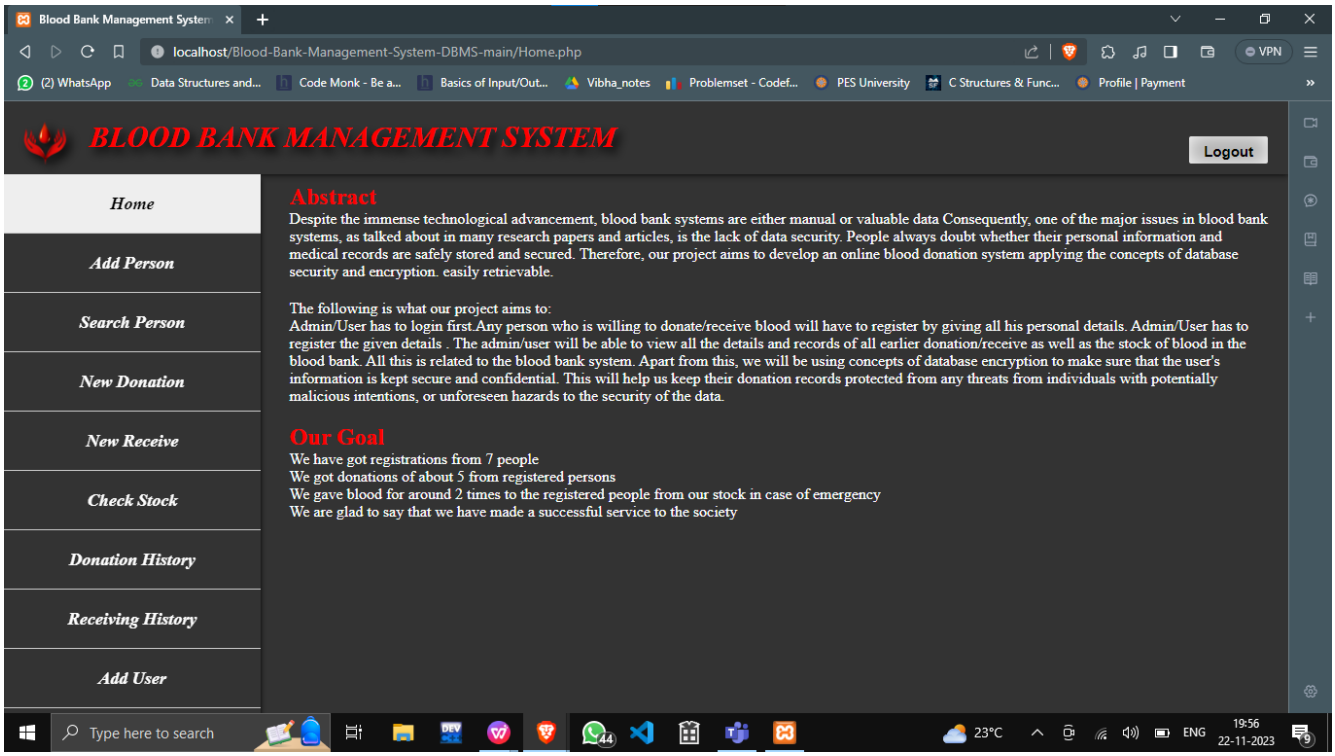


Fig 8.2: Home page

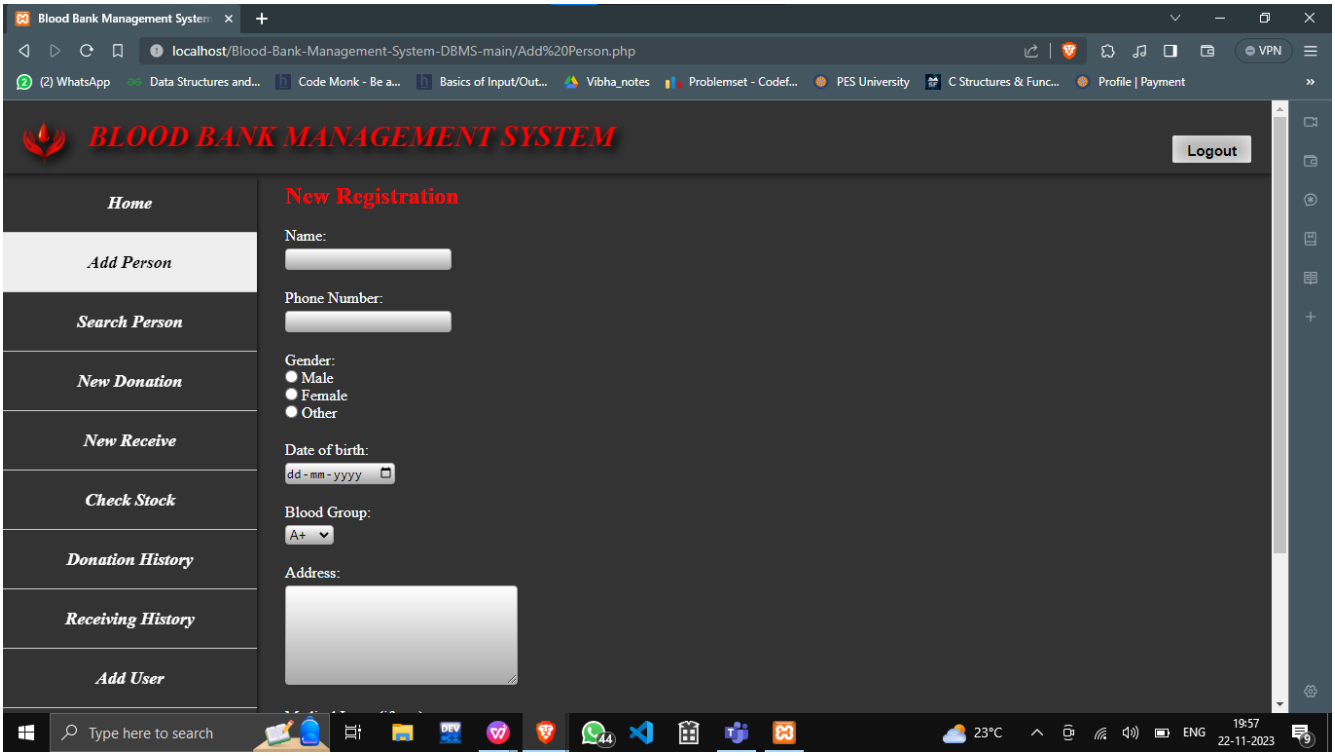


Fig 8.3: Add person page

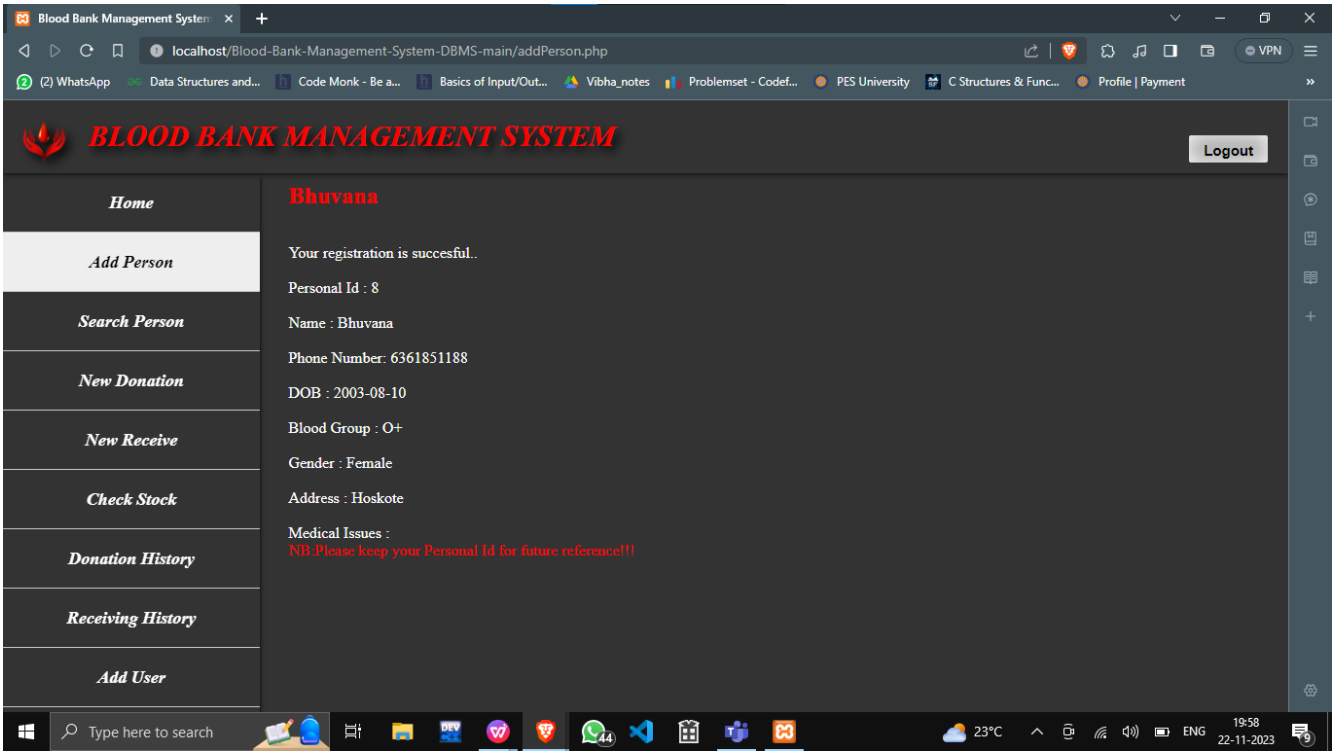


Fig 8.3.1: Add person page

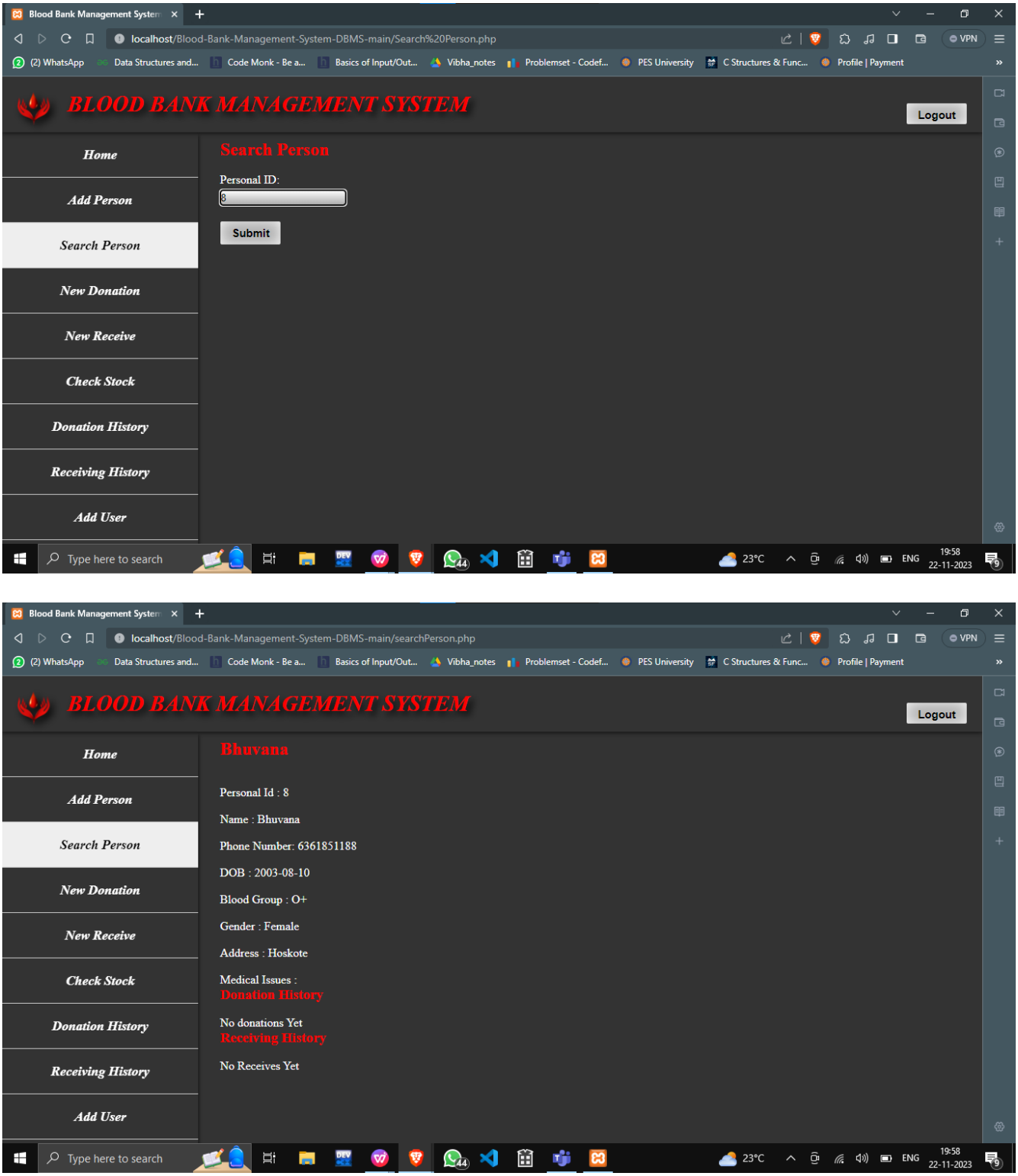


Fig 8.4: Search person page

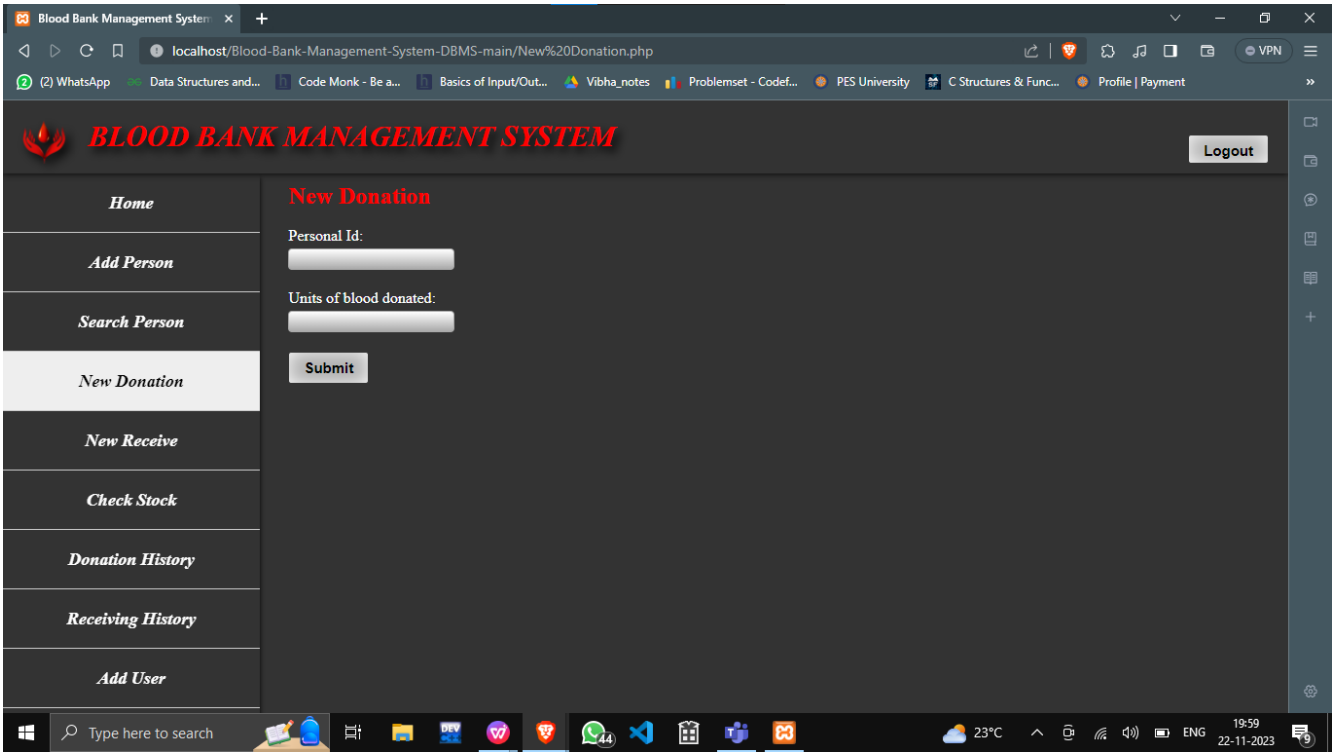


Fig 8.5: Donation page

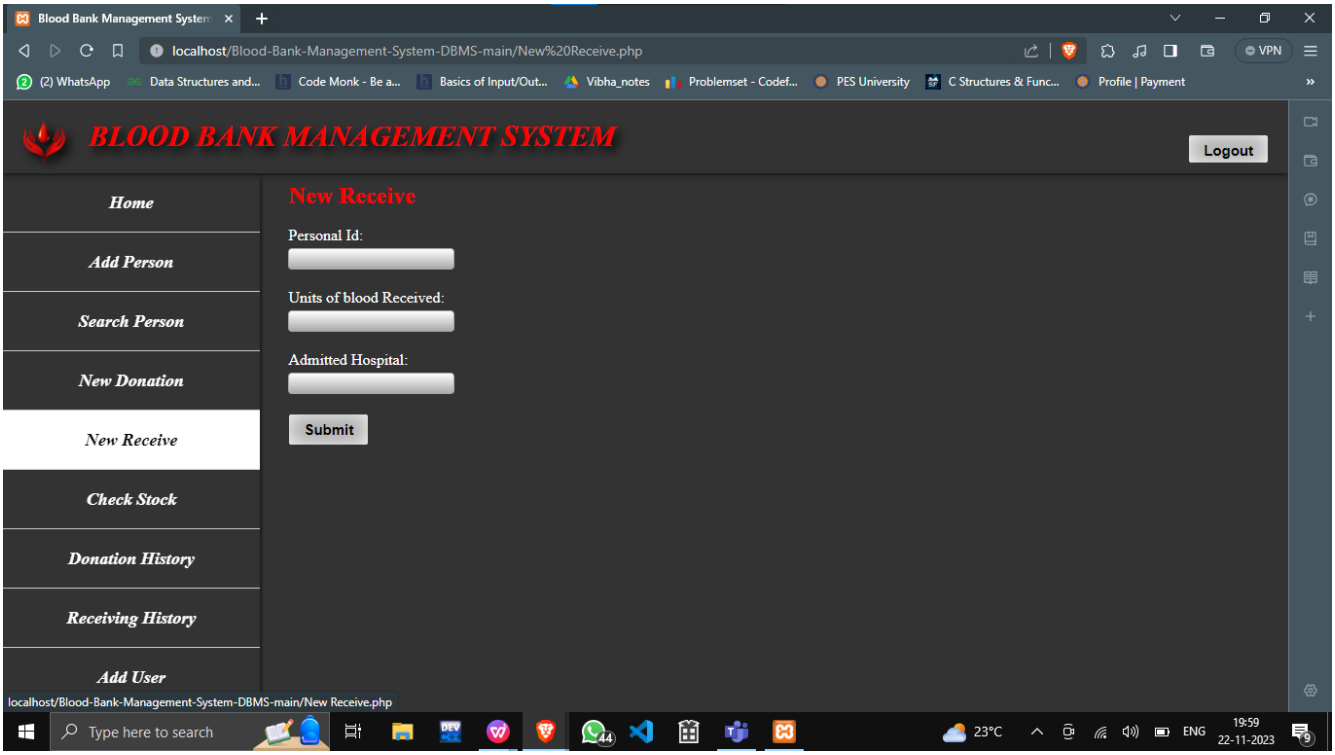


Fig 8.6: Receive page

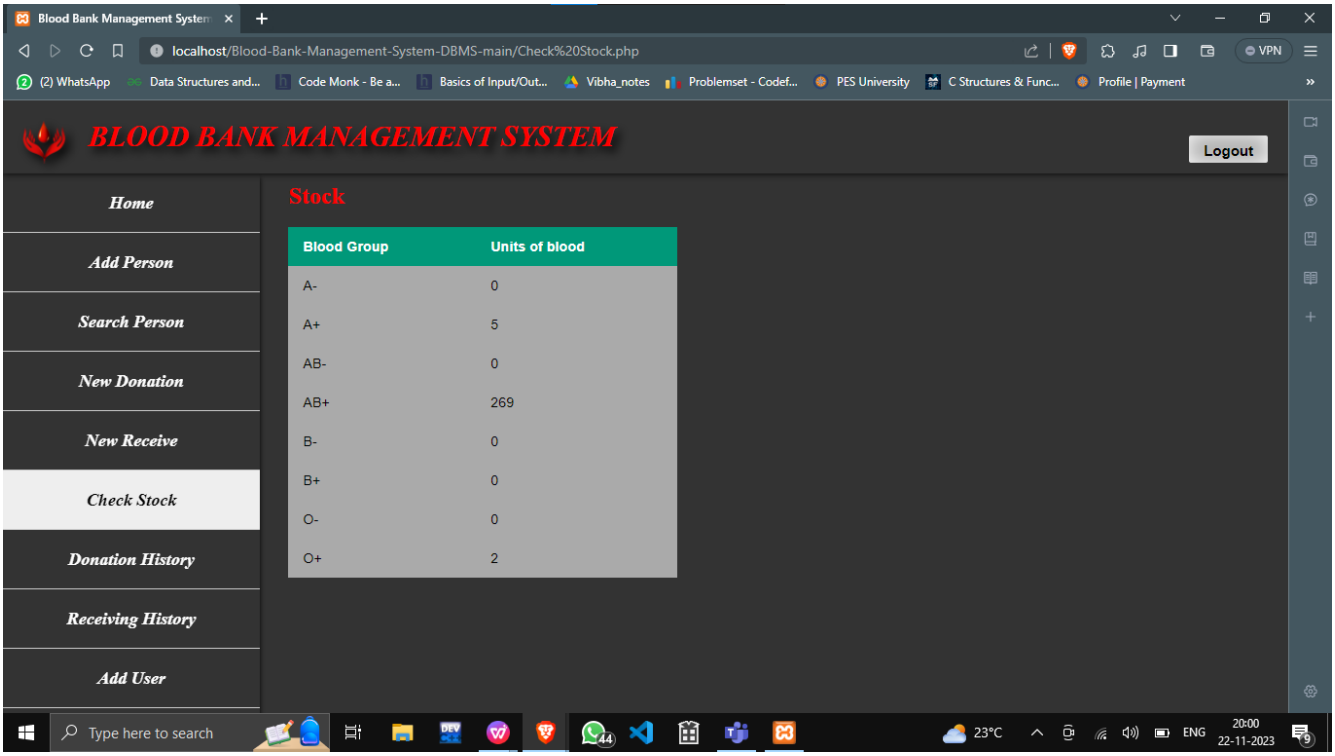


Fig 8.7: Check Stock page

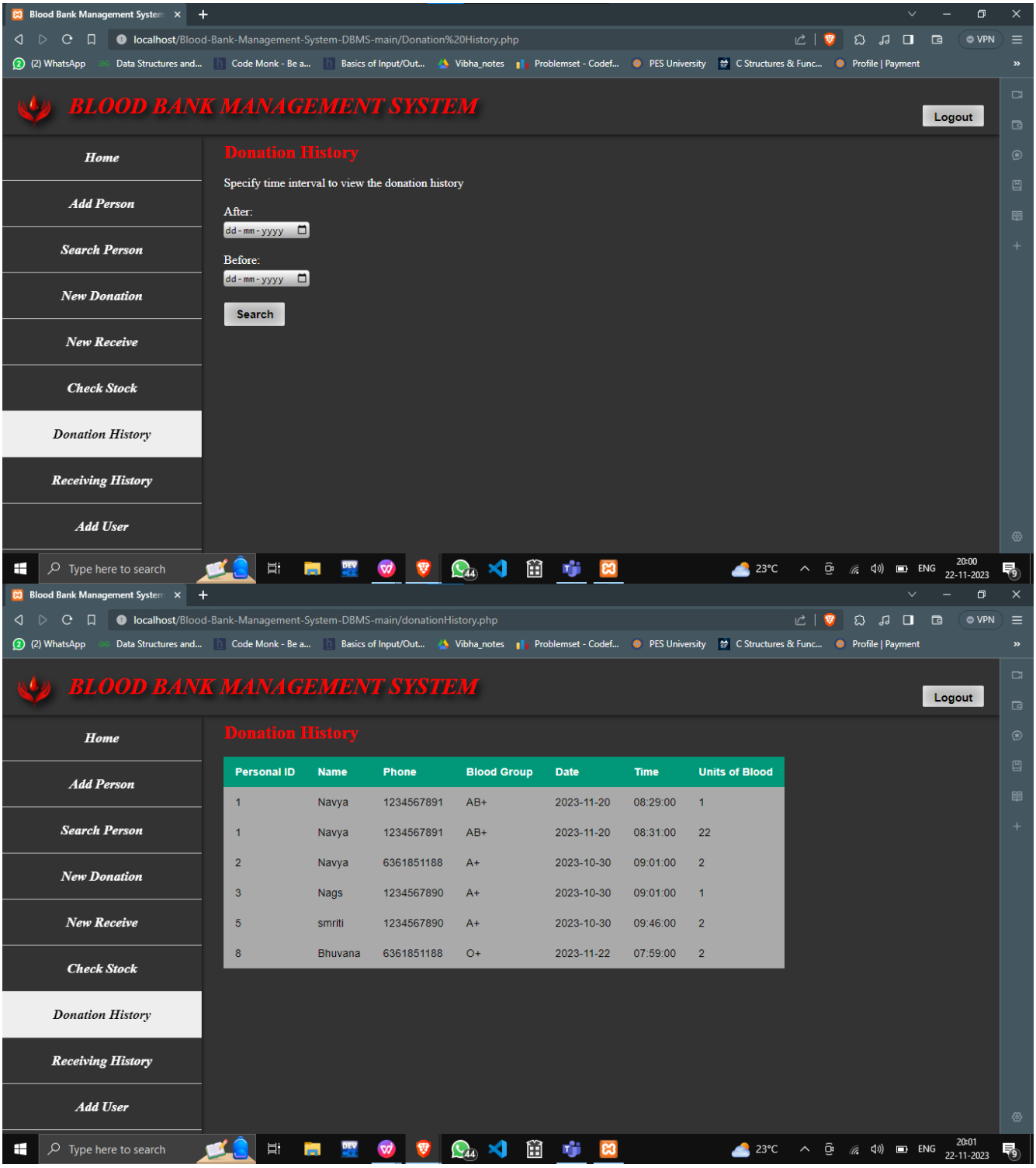
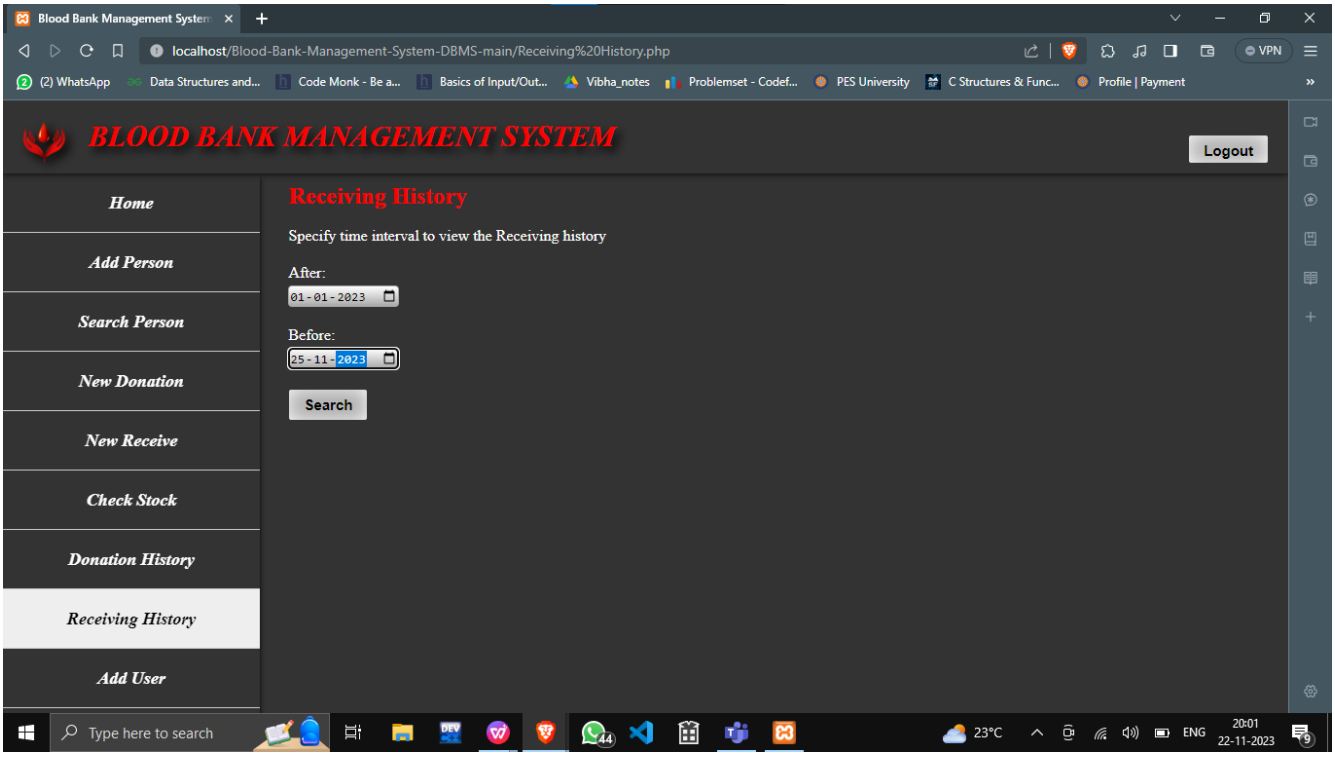


Fig 8.8: Donation History page



Z

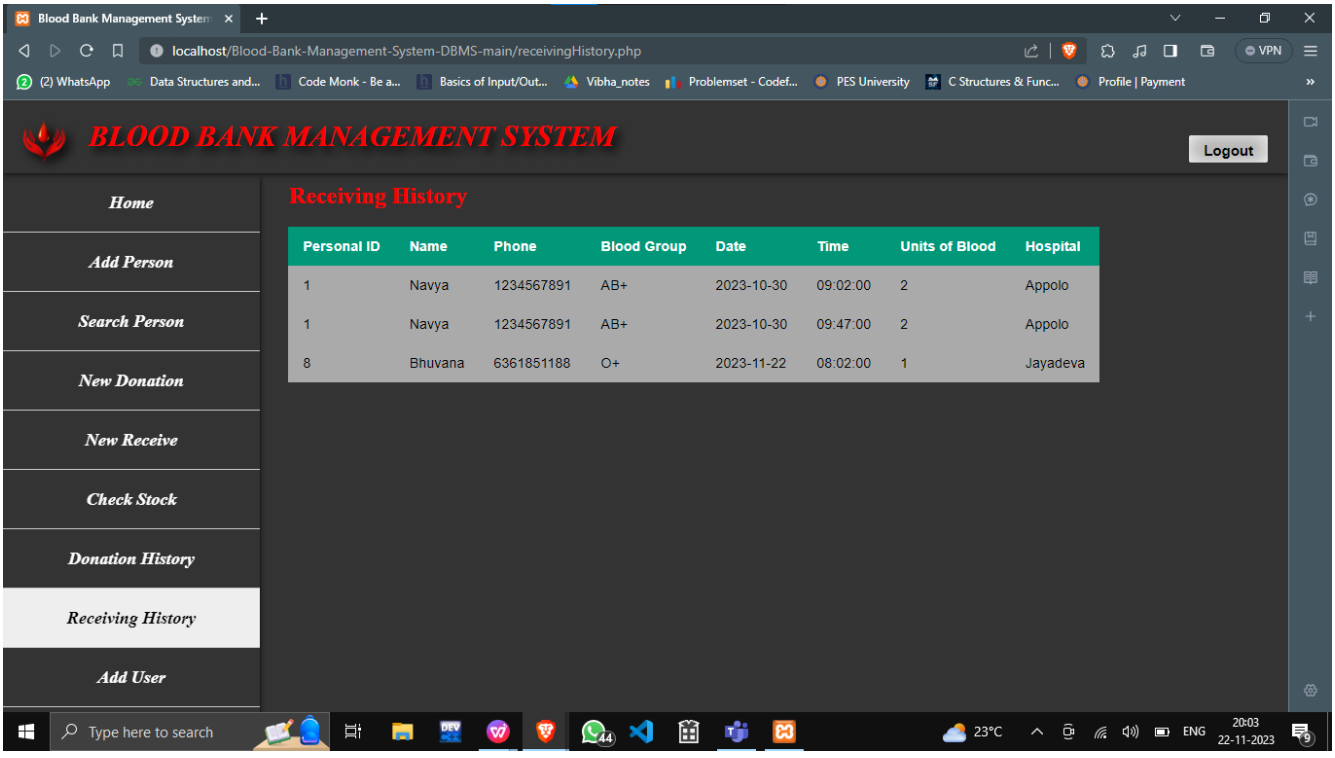


Fig 8.9: Receive History page

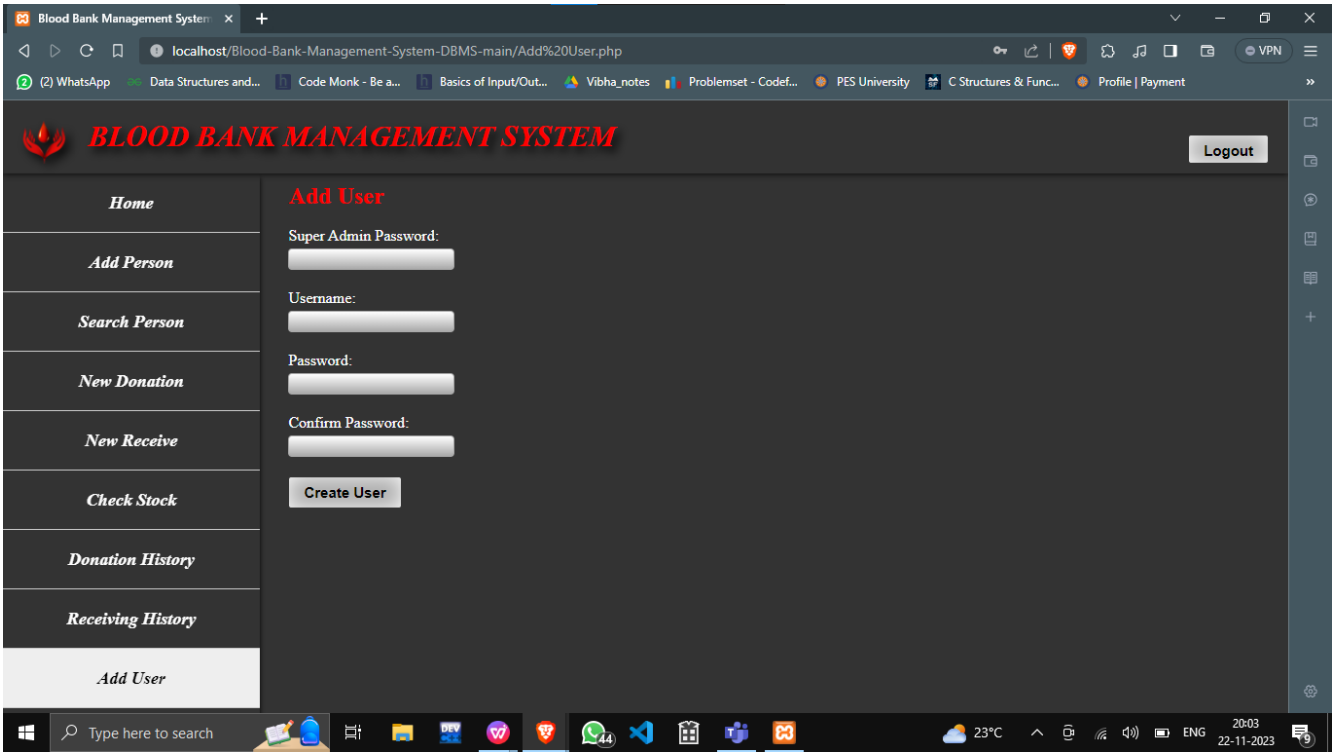


Fig 8.10: Add user page

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.
- Google
- https://www.w3schools.com