



UNIVERSITI TEKNIKAL MALAYSIA MELAKA
FAKULTI TEKNOLOGI MAKLUMAT DAN KOMUNIKASI

WORKSHOP I
REPORT

Name:	<u>NG WEI HEN</u>
Matric Number:	<u>B032110201</u>
Course:	<u>BITS S1G1</u>
Project Title:	<u>BLOOD DONATION MANAGEMENT</u> <u>SYSTEM</u>
Supervisor Name:	<u>TS. AZLIANOR BINTI ABDUL AZIZ</u>
Supervisor Signature:	<u></u>
Evaluator Name:	<u>TS. MUHAMMAD SUHAIZAN BIN</u> <u>SULONG</u>

Table of contents

No.	Contents	Page Number
1	Chapter 1: Introduction	1
	1.1 Problem Background	1
	1.2 Problem Statement	2
	1.3 Objectives	3
	1.4 Scopes	4
	1.5 Project Significance	5
2	Chapter 2: Analysis of Problem	6
	2.1 Problem Description	6
	2.2 Problem Decomposition	6
	2.3 Structured Chart	7
3	Chapter 3: Design	8
	3.1 Pseudo Code & Flowchart	8
	3.2 Entity-Relationship Diagram (ERD)	42
	3.3 Business Rule	43
	3.4 Data Dictionary	44
	3.5 Interface Design	47
4	Chapter 4: Implementation	80
	4.1 Introduction	80
	4.2 Data Manipulation Language (DML)	80
	4.3 Programming Language	82
	4.4 Selection	83
	4.5 Control	88
	4.6 Array	92
	4.7 Pointer	96
	4.8 Function	96
	4.9 Calculation	100
	4.10 Error Handling	101
5	Chapter 5: Conclusion	104
	5.1 Conclusion	104
	5.2 Suggestion	105
	References	106

CHAPTER 1 INTRODUCTION

1.1 Project Background

Blood donation is a process where a person willingly consents to having their blood collected and used for transfusions to the human in need so that they can overcome illnesses or cancer. The blood donation becomes an essential activity among humans as it can save many lives that are in need. The blood donation involves two individuals, these are donors and recipients. Different types of donors' blood correspond to different recipients as each of them is defined based on the presence of antigens on the surface red blood cells.

Nowadays, there are thousands of people that need donated blood so that their life can be sustained and stay healthy. Blood donation can help people based on many health conditions such as both internal and external bleeding, cancer treatment, inherited blood disorder and so on. Although there are a lot of pros, it is undoubtedly that there must be some cons too. In physical, people may face some minor skin problems like bruising or minor swelling of skin. They might also feel fatigued or dizzy as the blood pressure is temporarily decreasing. Hence people are advised to drink water before and after the donation. Of course, there are some conditions for a person that is not allowed to donate blood, for instance persons with health conditions such as cardiac disease, cancer and severe lung disease. A person with temporary health conditions such as pregnancy and acute fever is also prohibited from making blood donation.

Over the years, doctors and staffs have faced some hardships in managing the blood donation records as due to the data growth, lack of computerized system, and data duplication. In addition, the records are documented manually where the staff records blood donation details by just writing on paper. The handwriting might be illegible and unreadable over time as the handwriting has faded as well. Also, it is difficult and tedious to maintain the records as the records are messy and unorganized. Hence, it will be hard for both staff and donor to understand the donation details respectively. The records of donors and staff are not secured too as the manual management of data is

complicated and time-consuming. Hence the modern computerized system is ready to replace the manual-based techniques.

In this case, a simple blood donation management system is developed and designed to help the staff and donors manage the donation details that is easy but efficient way. Oracle SQL will be applied in this system as the centralized storage of database and retrieval of donor and staff records. Considering mobile gadgets are one of the mainstream gadgets for donors and staff nowadays, a system is implemented so that donors can report problems through the system. C++ would be the main interface of systems for admin, donors and staff to communicate.

1.2 Problem Statement

There are some issues that exist in current management. First, the difficulty of retrieving back the information related to blood donation between staff and donors. It could be hard to retrieve the record of a donor which has been created before if the donation details is updated. The staff has to manually search and find the record by going through one-by-one which results in wastage of time and efficiency. This process is not only time-consuming but also inefficient.

Risk of losing information of donors or staffs over time period. If the records are not saved and organized in a safety place, some records might be missing out and this leads to incompleteness of information. In addition, due to COVID-19 pandemic, many people have to make self-quarantine and hence all the documents related blood donation are placed unorganized. Hence, losing information might be a potential harm to the staffs and donors.

Difficulty in documentation for the records related to donors, staff, blood stock, blood donation and hospital. Without computerized methodology, staff have to record the details of staff, donors, blood donation, blood stock and hospital manually which is handwriting. Plus, the handwriting might be faded after a period of time. This causes

the documentation of records become even more tedious to be completed and results in time-consuming.

1.3 Objectives

This project embarks on the following objectives:

1. To enable tracking the some essentials details such as donors and staffs details. When staffs want to update the donation details for a person, they have to search the attribute of donors such as name, address, and also the donation details such as donation date and so on to find out the person's record. Hence, using computerised system, staff can filter or search the attribute to find the correspond records. The duration of finding records will be greatly reduced too. From donors' view, the donors will always be notified about the update of their own donation details and they are able to access so that they can always view the update of the details.
2. To maintain records of blood donors, blood donation information, staffs, hospitals, and blood stock details in a centralized database system. By doing so, staffs and donors can always Indirectly, this can also help the staff to keep tracking the donation records when the details are updated from time to time. For example, the system can help staffs to record the blood donation of a certain donor, so the staff will know whether the donor is qualified to make donation based on certain time or period such as more than 6 months or less than that. This could help staff to manage the information in a beneficial way.
3. To generate reports of the system based on preferences. They can view certain details based on the attributes such as donors personal details, blood donation records and so on. This is more convenient for donors and staffs as they want to view the details in an organized way for a better understanding.. The reports generated are saved in the database so that information loss will be prevented. The staffs and donors can filter the attribute to specify the report view based on the blood donation information. The report is integrated as it may help the staffs to make accounting such as total number

of donors, number of donation made yearly and monthly.

1.4 Scope

1. Developed Module

The system is mainly to be focused on some main modules, which are:

- Authentication & Validation modules
 - Login and Logout
 - Registration
 - Forget Password
- User Profile (Staff and Donor) Management
 - Add record
 - Display record
 - Delete record
 - Update record
 - Search record
- Blood Management
- Report generation for record
 - Blood donation
 - Donation details

2. Target User

The system is mainly to be used for three main users, which are:

- Admin
 - Login and logout
 - Create, update, delete, search the records of staff
 - Generate reports about staff records
- Staffs
 - Login and logout
 - Create, update, delete, and retrieve blood donation records.
 - Create, update, delete, and query donor's records in order to manage donor information.

- Create, update, delete, and query hospital's records in order to manage hospital information.
- Create, update, delete, query and retrieve blood stock records to manage blood information.
- View staff's own account information
- Generate report based on blood donation records
- Donors
 - Login and Logout
 - View the donors' donation record.
 - View donor's own account information
 - Will be provided the report for their own donation details.

1.5 Project Significance

The significance of this project is as following:

1. The system enables the staffs to search and view the details of blood donation for each pair of donor in an organised report view.
2. The staffs will be able to make any update and deletion for the staffs and donors information.
3. The system can provides graphical view such as command-line bar graph to display report of number of staffs based on certain attributes.
4. The system can increase the efficiency of arranging the details for blood donation as it reduces the cost and time to collect the data from donors and staffs.
5. Donors can easily get access to his or her own blood donation details.
6. The system is able to generate statistical reports on blood donation records such as calculating the number of records based on selected period.

CHAPTER 2 PROBLEM ANALYSIS

2.1 Problem Description

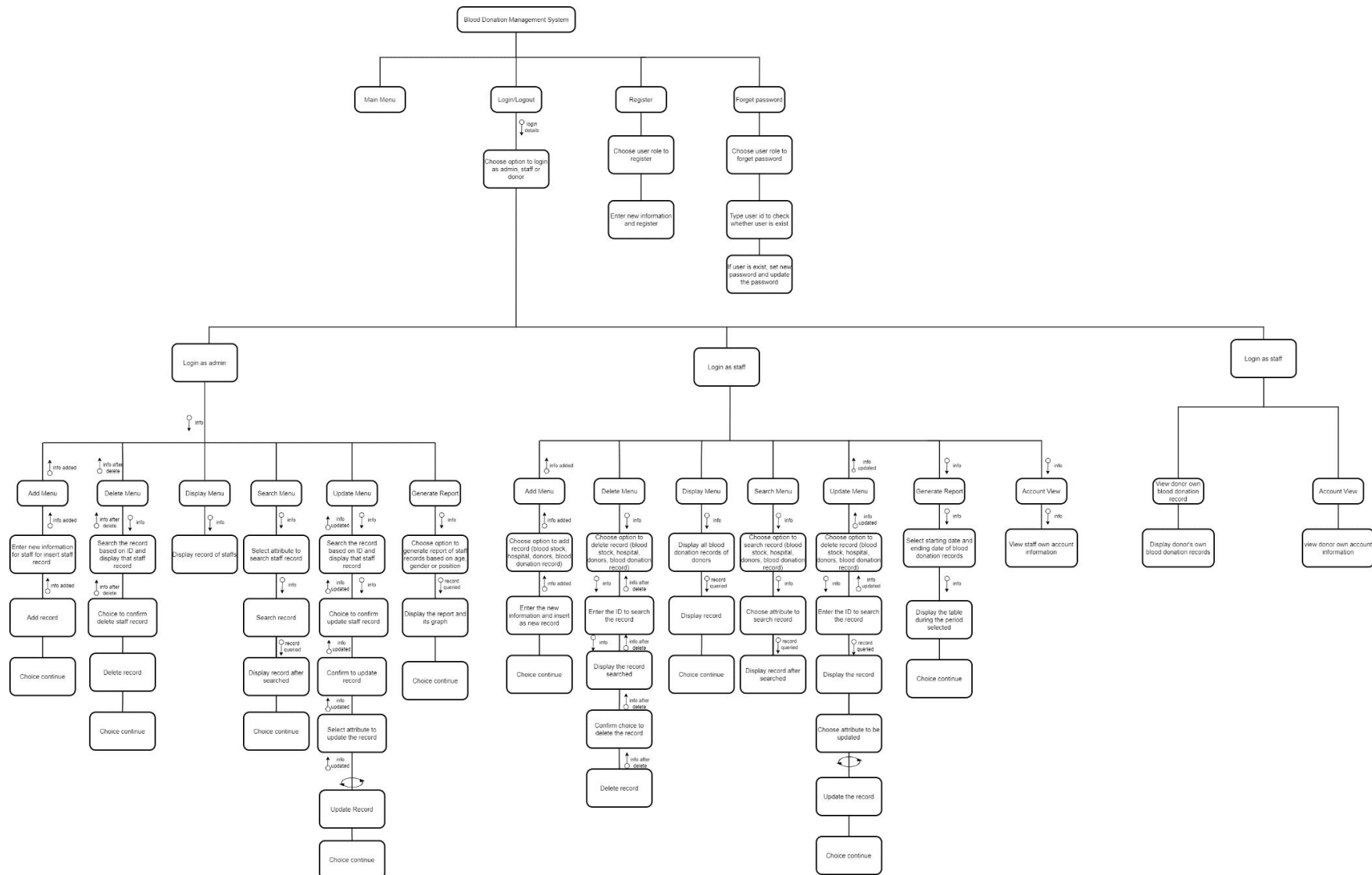
In the manual system, it is very troublesome to handle as there's limitation for managing the information about donation details. Administration and staffs have found that it will be time-consuming if they want to make manipulation on records such as adding, deleting, updating and so on as a lot of work load is required to do so. Plus, the accuracy of information might be reduced due to human error, such as miscounting the query and this might lead to the not tally of manual system. The staff might not be able to keep track the donation records if there is any error of donors' information.

In existing system, donors are unable to get know about their own personal donation records other than by having appointment with the counter physically. The problem will become worse if the personal information of donors are lost and cause unnecessary time-consuming to retrieve back the information again.

2.2 Problem Decomposition

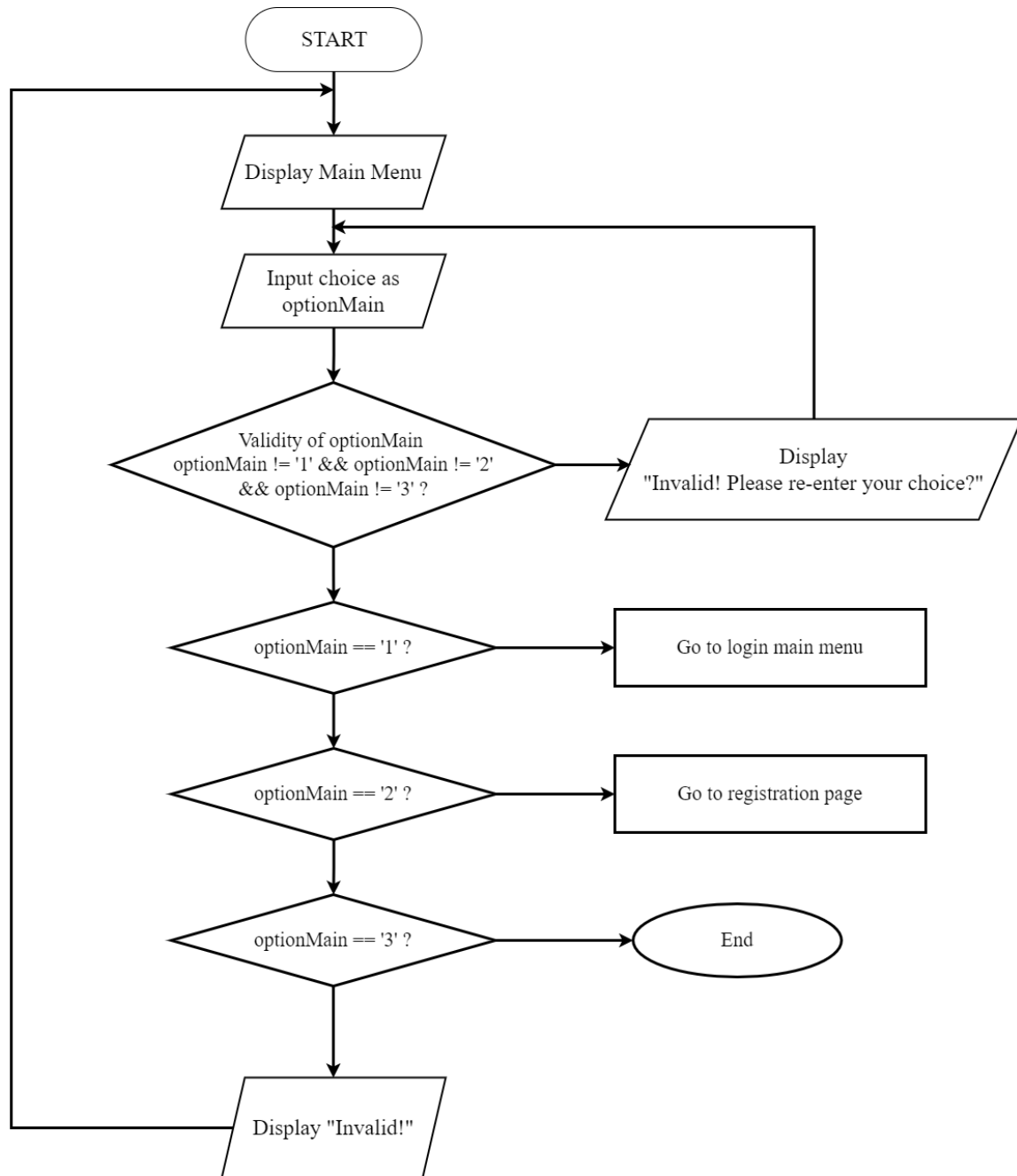
1. Database: Focuses on the sharing of various blood donation records in a large database
2. Report Listing: Provides a summarised view based on all the activities done by donors, staffs, blood donation, hospital and blood stock.
3. Search Engine: Enables staffs to filter the attribute such as donation date and search for any donation records
4. Authentication: Only admin are allowed to view the privileged information of all donors and staffs to prevent privacy of them from being leaked.

2.3 Structure Chart



3.1.1 Main Menu

Flowchart

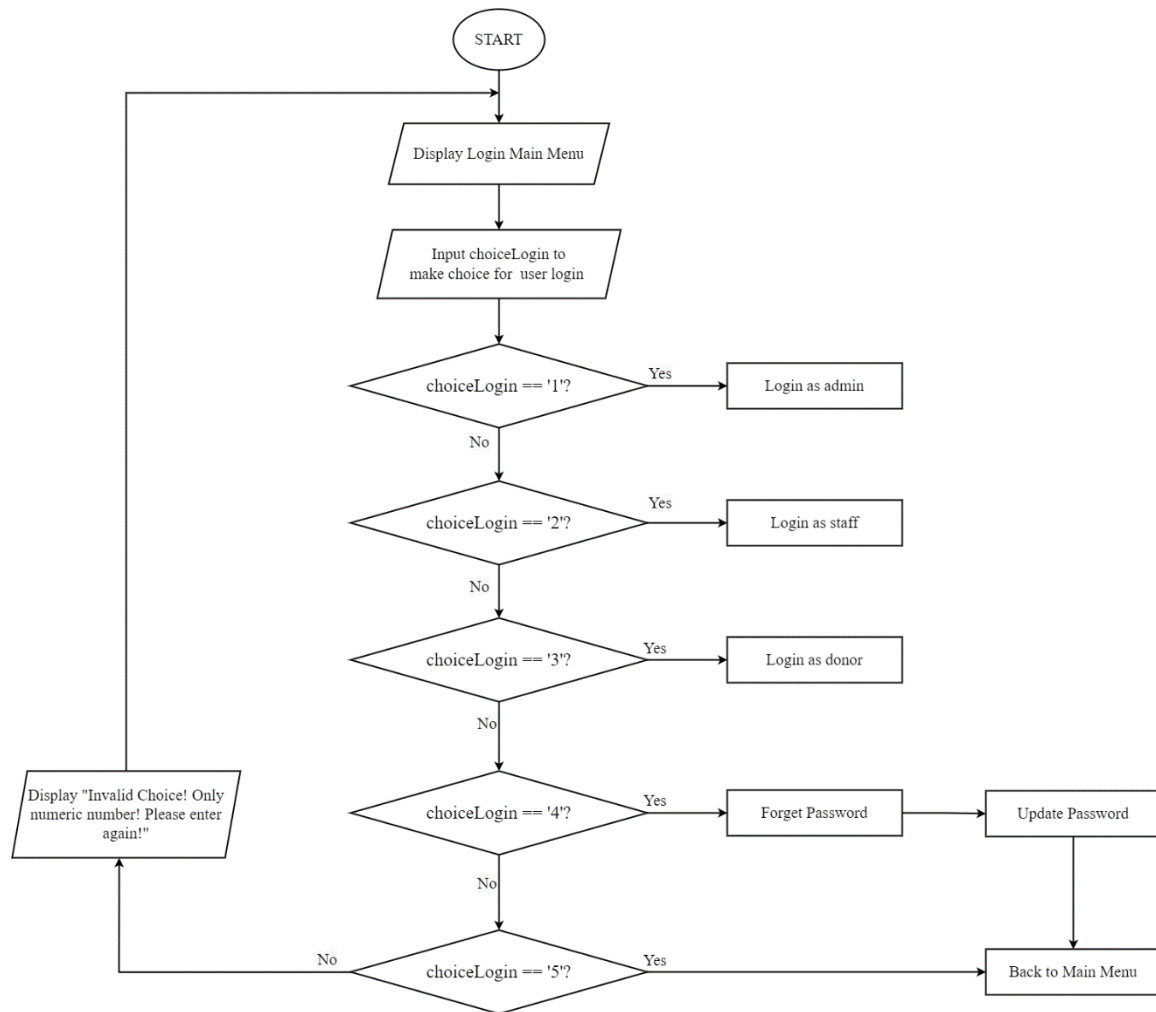


Pseudocode

1. START
2. Display Main Menu
3. Input OptionMain as choice
4. IF OptionMain is valid,
 - 4.1 Proceed to step 5ELSE Display “Invalid. Please re-enter your choice?”
 - 4.2 Back to step 3
5. IF the input OptionMain == ‘1’,
 - 5.1 Go to Login()ELSE IF the input OptionMain == ‘2’,
 - 5.2 Go to Registration()ELSE IF the input OptionMain == ‘3’,
 - 5.3 Exit the programELSE
 - 5.4 Display “Invalid”
 - 5.5 Back to step 2END IF
6. END

3.1.2 Login

Flowchart

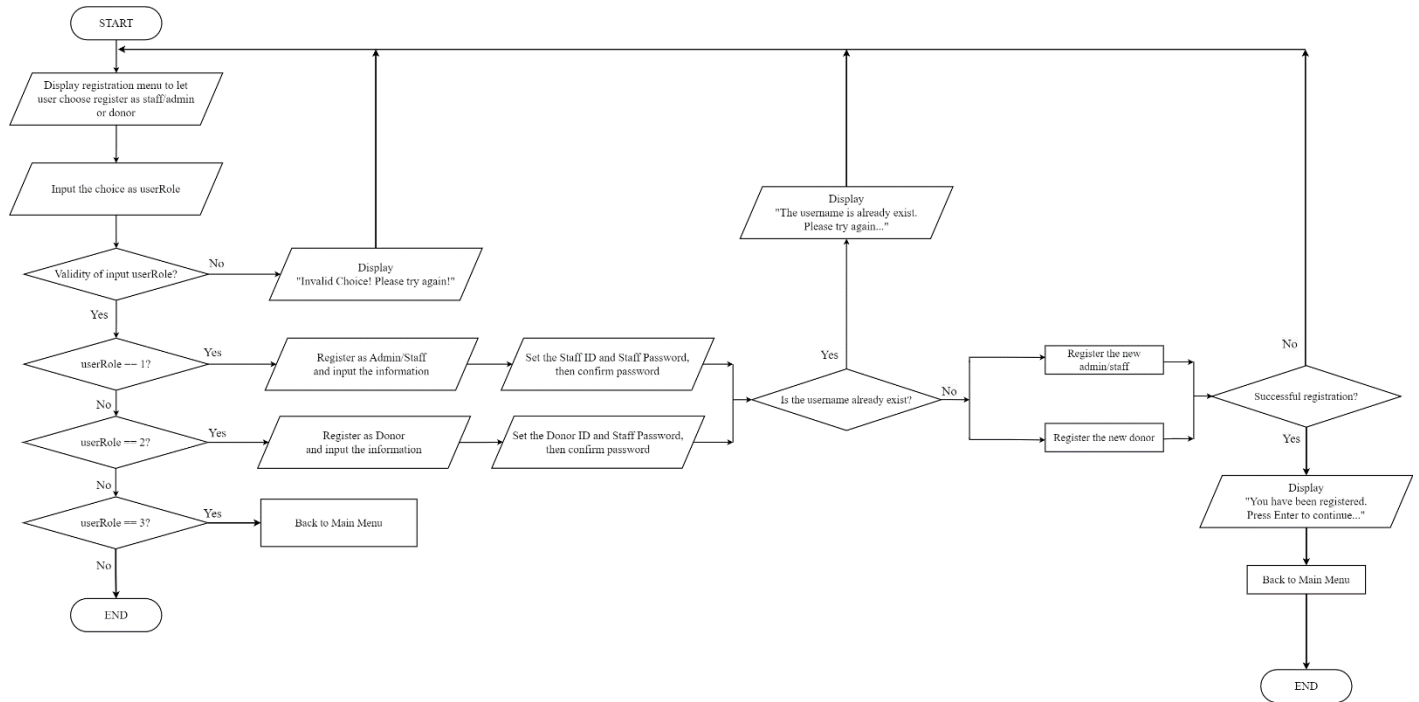


Pseudocode

1. START
2. Display Login Main Menu
3. Input choiceLogin
4. IF the input choiceLogin == '1',
 - 4.1 Go to AdminLogin()
- ELSE IF the input choiceLogin == '2',
 - 4.2 Go to StaffLogin()
- ELSE IF the input choiceLogin == '3',
 - 4.3 Go to DonorLogin()
- ELSE IF the input choiceLogin == '4',
 - 4.4 Go to ForgetPassword()
- ELSE IF the input choiceLogin == '5',
 - 4.5 Back to Main Menu
- ELSE
 - Display "Invalid Choice! Only numeric number! Please enter again!"
 - Back to step 3
- END IF
5. Log out
6. END

3.1.3 Registration

Flowchart

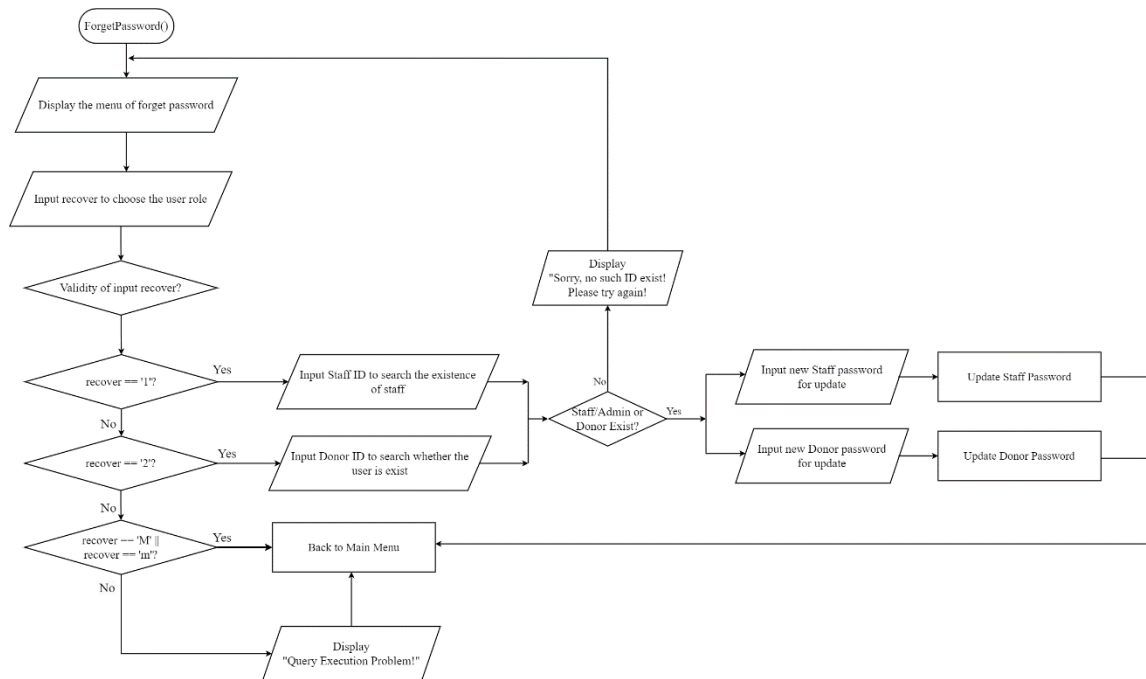


Pseudocode

1. START
2. Display registration menu to let user choose to register as staff/admin or donor
3. Input userRole as choice
4. IF input userRole valid,
 - 4.1 Proceed to step 5ELSE Display “Invalid. Please re-enter again!”
 - 4.2 Back to step 2
5. IF the input userRole == ‘1’,
 - 5.1 Register as admin or staff
 - 5.2 Input the information
 - 5.3 Proceed to step 6ELSE IF the input userRole == ‘2’,
 - 5.4 Register as donor
 - 5.5 Input the information
 - 5.6 Proceed to step 6ELSE IF the input choiceLogin == ‘3’,
 - 5.7 Back to Main MenuELSE
 - Display “Invalid Choice! Only numeric number! Please enter again!”
 - Back to step 3END IF
6. Log out
7. END

3.1.4 Forget Password

Flowchart



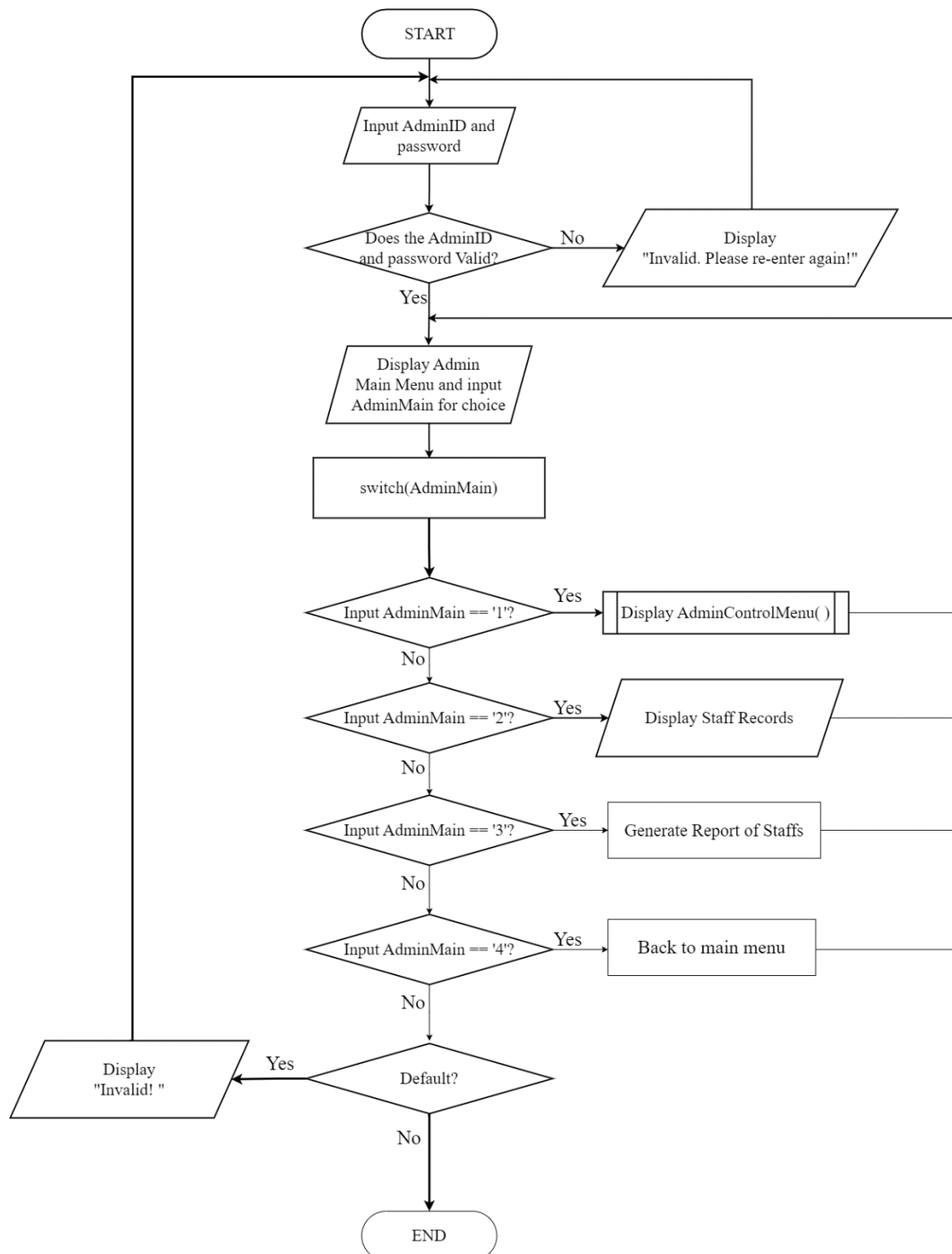
Pseudocode

1. START
2. Display menu of forget password
3. Input recover to choose user role
4. IF input recover valid,
 - 4.1 Proceed to step 5
 - ELSE Display “Invalid. Please try again!”
 - 4.2 Back to step 2
- 5 IF the input recover == ‘1’,
 - 5.1 Input Staff_ID to search existence of staff
 - 5.2 IF staff record is found
 - 5.2.1 Input new staff password
 - 5.2.2 Update new staff password
 - 5.2.3 Proceed to step 6

```
        ELSE
            5.2.4  Display "Sorry, no such ID exist! Please try again!"
            5.2.5  Back to step 2
        ELSE IF the input recover == '2',
            5.3  Input Donor_ID to search existence of staff
            5.4  IF donor record is found
                5.4.1 Input new donor password
                5.4.2 Update new donor password
                5.4.3 Proceed to step 6
            ELSE
                5.4.4  Display "Sorry, no such ID exist! Please try again!"
                5.4.5  Back to step 2
        ELSE IF the input recover == 'M' || recover == 'm',
            5.5 Back to Main Menu
        ELSE
            5.6 Display "Invalid Choice! Only numeric number! Please enter again!"
            5.7 Back to step 3
        END IF
5  Log out
6  END
```

3.1.5 Admin Main Menu

Flowchart

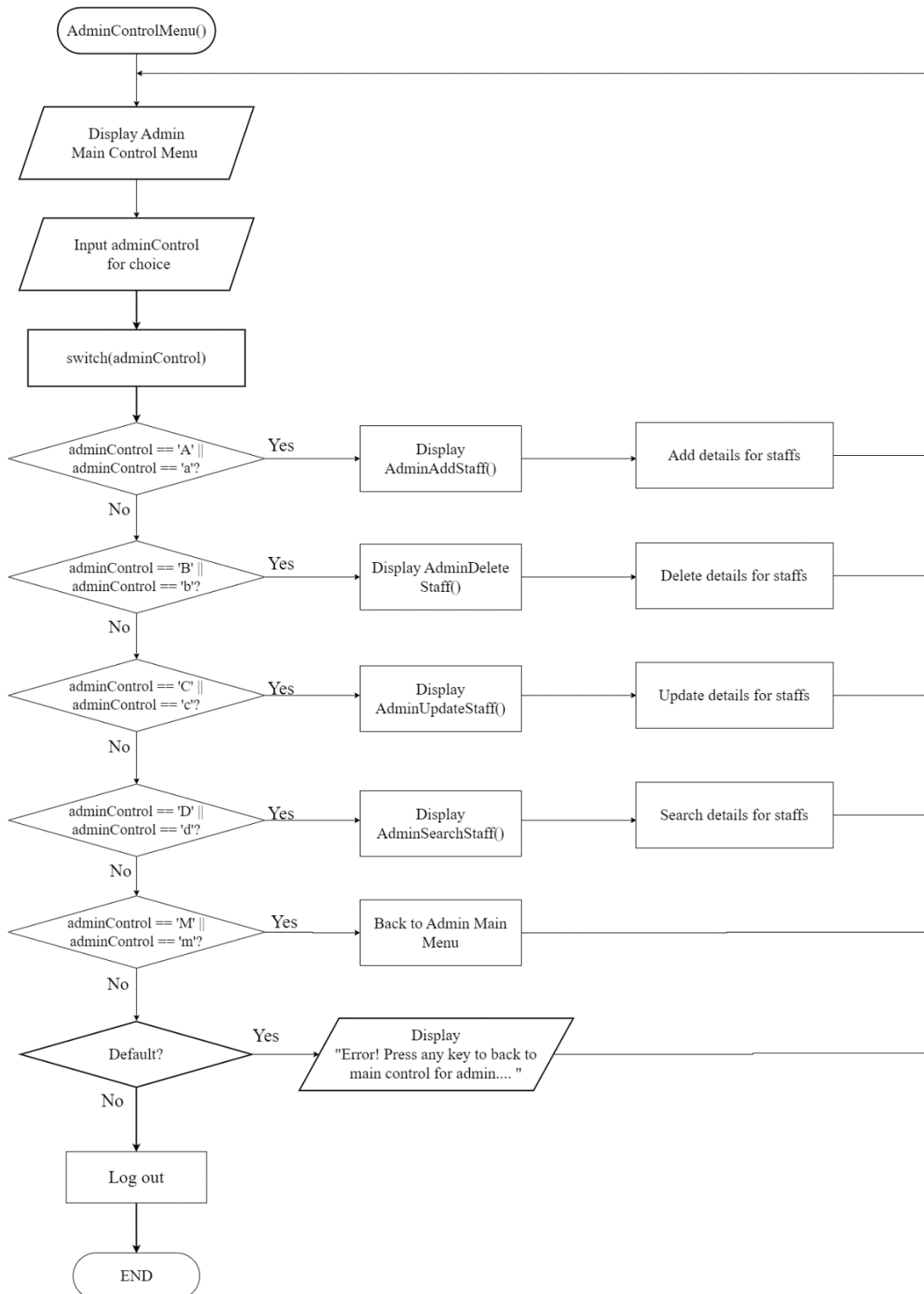


Pseudocode

1. START
2. Input AdminID and Password
3. IF the AdminID and Password are valid,
 - 3.1 Proceed to step 4ELSE Display “Invalid. Please re-enter again!”
 - 3.2 Back to step 2
4. Display Admin Main Menu and input AdminMain for choice
5. IF the input AdminMain is valid,
 - 5.1 Proceed to step 6ELSE Display “Invalid choice!”
 - 5.2 Back to step 4
6. Switch(AdminMain)
7. IF the input AdminMain == ‘1’,
 - 7.1 Call out function AdminControlMenu()
 - 7.2 Back to step 4ELSE IF the input AdminMain == ‘2’,
 - 7.3 Display Staff Records
 - 7.4 Back to step 4ELSE IF the input AdminMain == ‘3’,
 - 7.5 Generate Report of Staffs
 - 7.6 Back to step 4ELSE IF the input AdminMain == ‘4’,
 - 7.7 Back to Admin Main Menu
 - 7.8 Back to step 4ELSE Proceed to step 8
8. END IF
9. Log out
10. END

3.1.6 Admin Control Menu

Flowchart

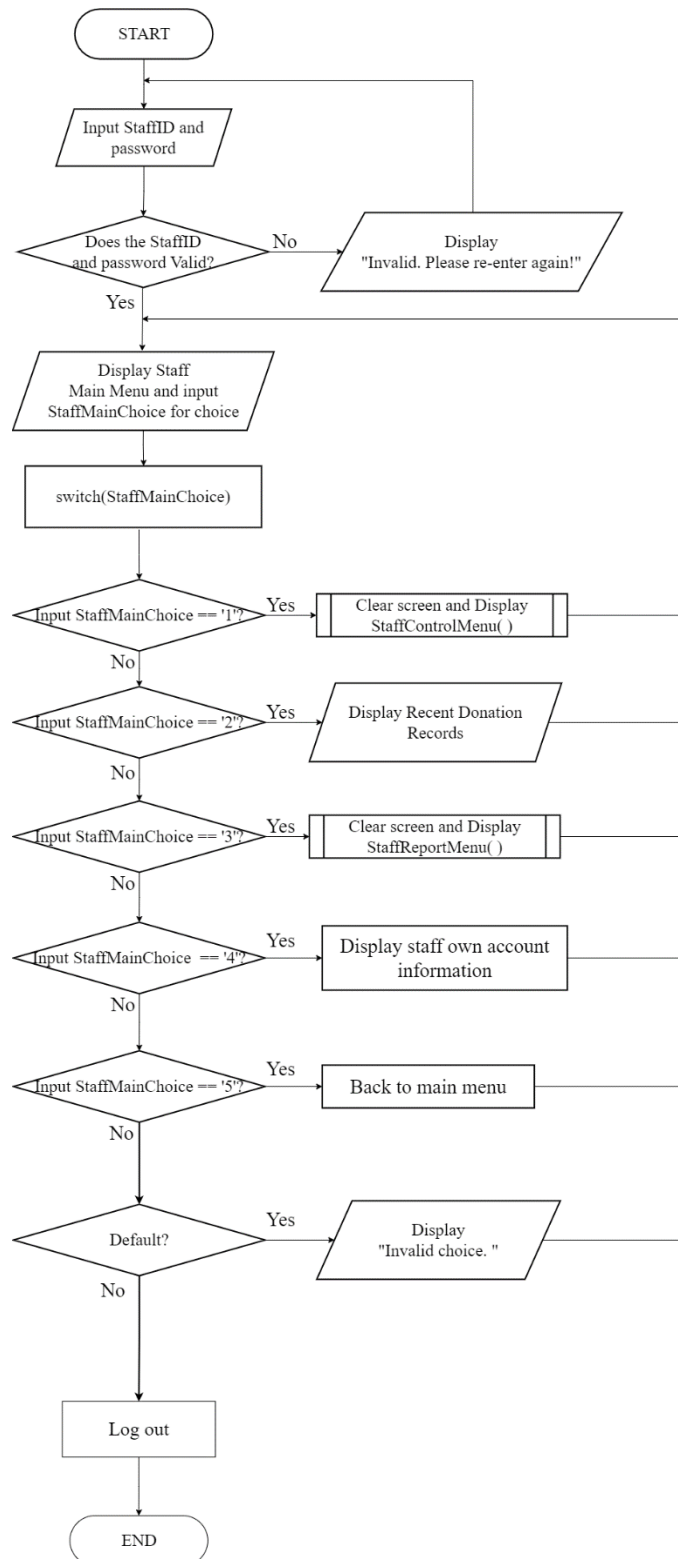


Pseudocode

1. START
2. Display Admin Main Control Menu
3. Input adminControl for choice
4. Switch(adminControl)
5. IF the input adminControl == 'A' || adminControl == 'a',
 - 5.1 Display AdminAddStaff()
 - 5.2 Add details for staffs
 - 5.3 Proceed back to step 2
- ELSE IF the input adminControl == 'B' || adminControl == 'b',
 - 5.4 Display AdminDeleteStaff()
 - 5.5 Delete details for staffs
 - 5.6 Proceed back to step 2
- ELSE IF the input adminControl == 'C' || adminControl == 'c',
 - 5.7 Display AdminUpdateStaff()
 - 5.8 Update details for staffs
 - 5.9 Proceed back to step 2
- ELSE IF the input adminControl == 'D' || adminControl == 'd',
 - 5.10 Display AdminSearchStaff()
 - 5.11 Search details for staffs
 - 5.12 Proceed back to step 2
- ELSE IF the input adminControl == 'M' || adminControl == 'm',
 - 5.13 Back to Admin Main Menu
- ELSE Proceed to step 6
- END IF
6. Log out
7. END

3.1.7 Staff Main Menu

Flowchart

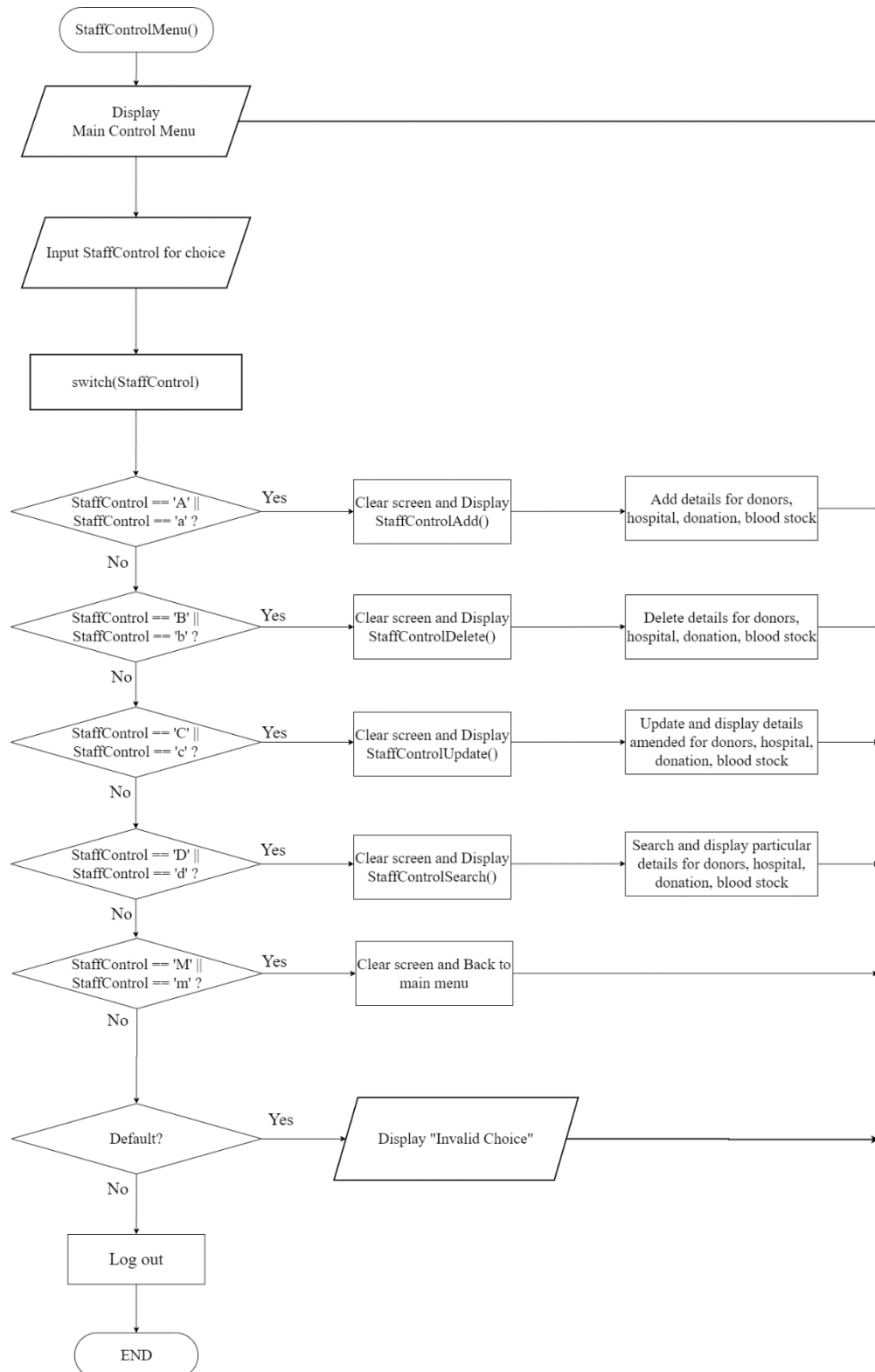


Pseudocode

1. START
2. Input StaffID and Password
3. IF the StaffID and Password are valid,
 - 3.1 Proceed to step 4ELSE Display “Invalid. Please re-enter again!”
 - 3.2 Back to step 2
4. Display Staff Main Menu and input StaffMainChoice for choice
5. IF the input StaffMainChoice is valid,
 - 5.1 Proceed to step 6ELSE Display “Invalid choice!”
 - 5.2 Back to step 4
6. Switch(StaffMainChoice)
7. IF the input StaffMainChoice == ‘1’,
 - 7.1 Call out function StaffMainChoice ()
 - 7.2 Back to step 4ELSE IF the input StaffMainChoice == ‘2’,
 - 7.3 Display Recent Donation Records
 - 7.4 Back to step 4ELSE IF the input StaffMainChoice == ‘3’,
 - 7.5 Generate Report
 - 7.6 Back to step 4ELSE IF the input StaffMainChoice == ‘4’,
 - 7.7 Display Staff own account information
 - 7.8 Back to step 4ELSE IF the input StaffMainChoice == ‘5’,
 - 7.9 Back to Staff Main MenuELSE Proceed to step 8
8. Log out
9. END

3.1.8 Staff Control Menu

Flowchart

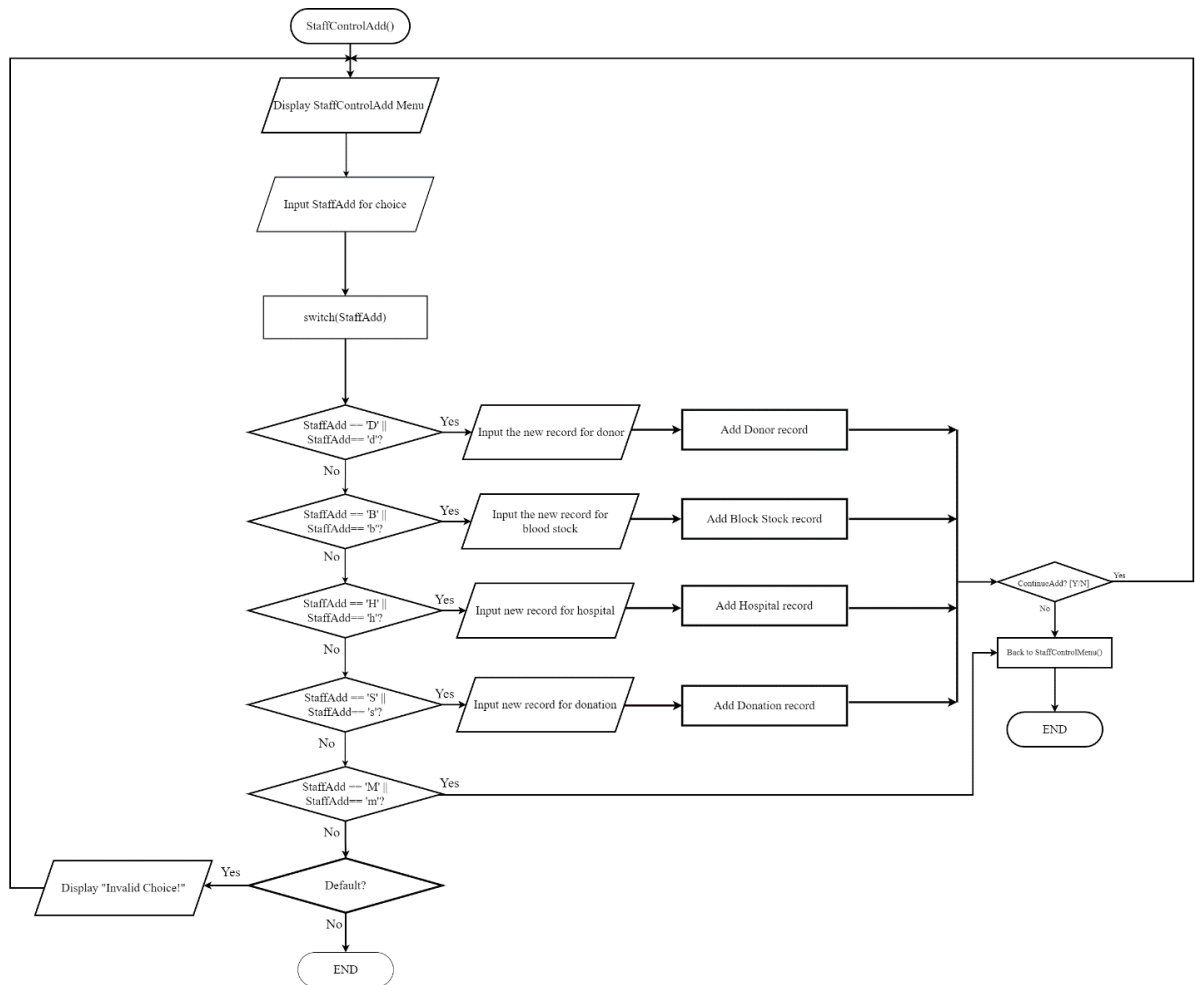


Pseudocode

1. START
2. Display Staff Main Control Menu
3. Input StaffControl for choice
4. Switch(StaffControl)
5. IF the input StaffControl == 'A' || input StaffControl == 'a',
 - 5.1 Display StaffControlAdd()
 - 5.2 Add details for donors, hospital, donation, blood stock
 - 5.3 Proceed back to step 2
- ELSE IF the input StaffControl == 'B' || input StaffControl == 'b',
 - 5.4 Display StaffControlDelete()
 - 5.5 Delete details for donors, hospital, donation, blood stock
 - 5.6 Proceed back to step 2
- ELSE IF the input StaffControl == 'C' || input StaffControl == 'c',
 - 5.7 Display StaffControlUpdate()
 - 5.8 Update details for donors, hospital, donation, blood stock
 - 5.9 Proceed back to step 2
- ELSE IF the input StaffControl == 'D' || input StaffControl == 'd',
 - 5.10 Display StaffControlSearch()
 - 5.11 Search details for donors, hospital, donation, blood stock
 - 5.12 Proceed back to step 2
- ELSE IF the input StaffControl == 'M' || input StaffControl == 'm',
 - 5.13 Back to StaffMainMenu()
- ELSE Proceed to step 6
- END IF
6. Log out
7. END

3.1.9 Staff Control Menu – Add

Flowchart



Pseudocode

1. Start
2. Display Staff control add menu
3. Input StaffAdd for choice
4. Switch (StaffAdd)
 5. IF (StaffAdd == 'D' || StaffAdd == 'd')
 - 5.1 Input new record of donors
 - 5.2 IF user continue to add record
 - 5.2.1 Back to step 2

ELSE

5.2.2 Back to main Menu

ELSE IF (StaffAdd == 'H' || StaffAdd == 'h')

5.3 Input new record of hospital

5.4 IF user continue to add record

5.4.1 Back to step 2

ELSE

5.4.2 Back to main Menu

ELSE IF (StaffAdd == 'B' || StaffAdd == 'b')

5.5 Input new record of blood stock

5.6 IF user continue to add record

5.6.1 Back to step 2

ELSE

5.6.2 Back to main Menu

ELSE IF (StaffAdd == 'S' || StaffAdd == 's')

5.7 Input new record of blood donation

5.8 IF user continue to add record

5.8.1 Back to step 2

ELSE

5.8.2 Back to main Menu

ELSE IF (StaffAdd == 'M' || StaffAdd == 'm')

5.9 Back to Main Menu

ELSE

5.10 Display "Invalid Choice!"

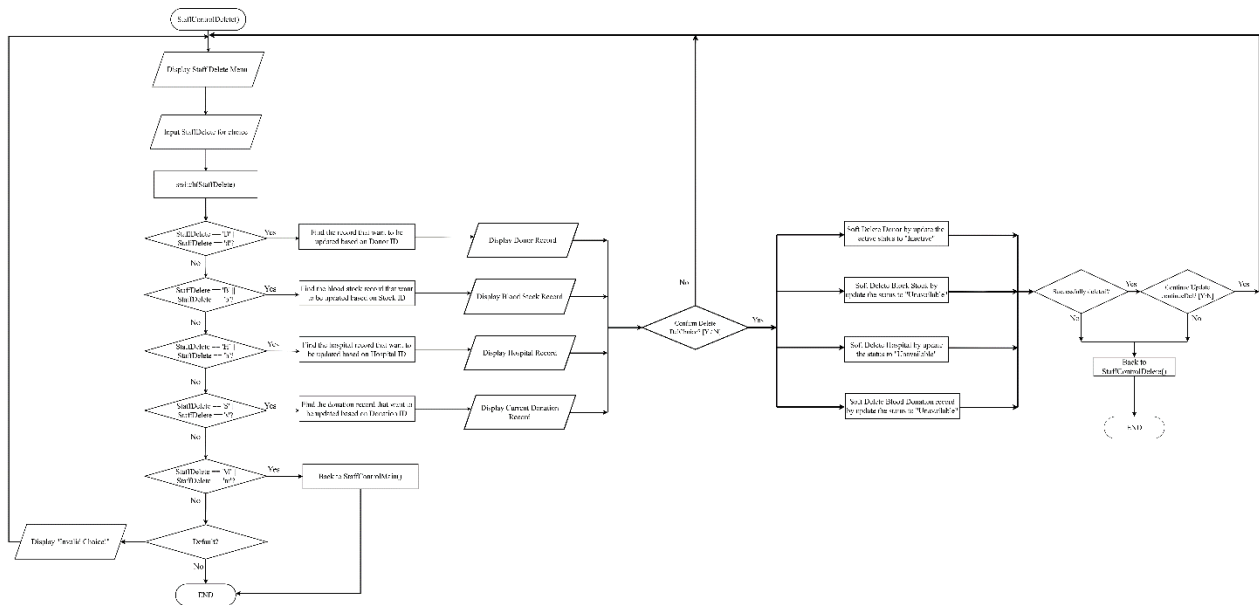
5.11 Back to step 2

END IF

6. END

3.1.10 Staff Control Menu – Delete

Flowchart



Pseudocode

1. Start
2. Display Staff control delete menu
3. Input StaffDelete for choice
4. Switch (StaffDelete)
5. IF (StaffDelete == 'D' || StaffDelete == 'd')
 - 5.1 Search donor record based on donor ID
 - 5.2 IF record found,
 - 5.2.1 IF confirm delete record
 - Soft delete Donor Record by updating active status to 'Inactive'
 - IF Successful delete
 - IF Confirm continue to delete donor record
 - Back to step 2
 - ELSE
 - Back to staff control menu
 - ELSE
 - Back to staff control menu

```
        Display "Query Execution Problem"
        Back to StaffControlDelete()
    ELSE
        Display "Query Execution Problem"
        Back to StaffControlDelete()

ELSE IF (StaffDelete == 'H' || StaffDelete == 'h')
    5.3 Search hospital record based on hospital ID
    5.4 IF record found,
        5.4.1 IF confirm delete record
            Soft delete hospital Record by updating status to 'Unavailable'
            IF Successful delete
                IF Confirm continue to delete hospital record
                    Back to step 2
            ELSE
                Back to staff control menu
        ELSE
            Display "Query Execution Problem"
            Back to StaffControlDelete()
    ELSE
        Display "Query Execution Problem"
        Back to StaffControlDelete()

ELSE IF (StaffDelete == 'B' || StaffDelete == 'b')
    5.5 Search blood stock record based on stock ID
    5.6 IF record found,
        5.6.1 IF confirm delete record
            Soft delete blood stock Record by update status to 'Unavailable'
            IF Successful delete
                IF Confirm continue to delete blood stock record
                    Back to step 2
```

```
        ELSE
            Back to staff control menu
        ELSE
            Display "Query Execution Problem"
            Back to StaffControlDelete()
        ELSE
            Display "Query Execution Problem"
            Back to StaffControlDelete()

ELSE IF (StaffDelete == 'S' || StaffDelete == 's')
    5.7 Search blood donation record based on stock ID
    5.8 IF record found,
        5.8.1 IF confirm delete record
            Soft delete blood donation record by updating status to 'Incomplete'
            IF Successful delete
                IF Confirm continue to delete blood donation record
                    Back to step 2
                ELSE
                    Back to staff control menu
            ELSE
                Display "Query Execution Problem"
                Back to StaffControlDelete()
        ELSE
            Display "Query Execution Problem"
            Back to StaffControlDelete()

ELSE IF (StaffDelete == 'M' || StaffDelete == 'm')
    5.9 Back to Main Menu

ELSE
    5.10 Display "Invalid Choice!"
```

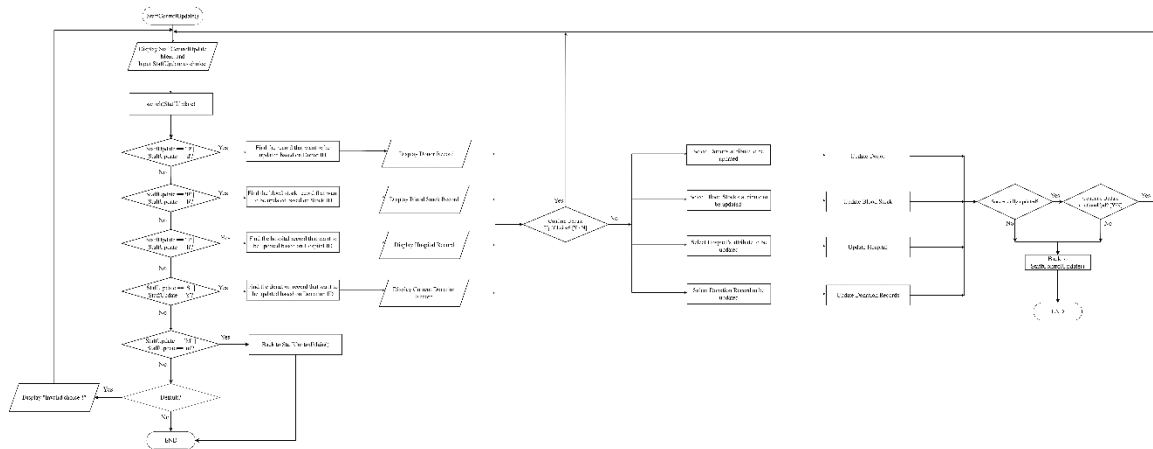
5.11 Back to step 2

END IF

6. END

3.1.11 Staff Control Menu – Update

Flowchart



Pseudocode

1. Start
2. Display Staff control Update menu
3. Input StaffUpdate for choice
4. Switch (StaffUpdate)
 5. IF (StaffUpdate == 'D' || StaffUpdate == 'd')
 - 5.1 Search donor record that wants to be updated based on donor ID
 - 5.2 IF record found,
 - 5.2.1 IF confirm update record
 - Select attribute of donor to update
 - Input new information for selected attribute to be updated
 - Update Donor Record
 - IF Successful update
 - IF Confirm continue to update donor record
 - Back to step 2
 - ELSE
 - Back to staff control menu
 - ELSE
 - Display "Query Execution Problem"

```
        Back to StaffControlUpdate()
    ELSE
        Display "Query Execution Problem"
        Back to StaffControlUpdate()

ELSE IF (StaffUpdate == 'H' || StaffUpdate == 'h')
    5.3 Search hospital record that wants to be updated based on hospital ID
    5.4 IF record found,
        5.4.1 IF confirm update record
            Select attribute of hospital to update
            Input new information for selected attribute to be updated
            Update hospital Record
            IF Successful hospital
                IF Confirm continue to update hospital record
                    Back to step 2
            ELSE
                Back to staff control menu
        ELSE
            Display "Query Execution Problem"
            Back to StaffControlUpdate()
    ELSE
        Display "Query Execution Problem"
        Back to StaffControlUpdate()

ELSE IF (StaffUpdate == 'B' || StaffUpdate == 'b')
    5.5 Search blood stock record that wants to be updated based on stock ID
    5.6 IF record found,
        5.6.1 IF confirm update record
            Select attribute of blood stock to update
            Input new information for selected attribute to be updated
            Update blood stock Record
```

```
IF Successful hospital
    IF Confirm continue to update hospital record
        Back to step 2
    ELSE
        Back to staff control menu
ELSE
    Display "Query Execution Problem"
    Back to StaffControlUpdate()
ELSE
    Display "Query Execution Problem"
    Back to StaffControlUpdate()

ELSE IF (StaffUpdate == 'S' || StaffUpdate == 's')
    5.7 Search blood donation record that wants to be updated based on donation ID
    5.8 IF record found,
        1.1.1 IF confirm update record
            Select attribute of blood donation to update
            Input new information for selected attribute to be updated
            Update blood donation Record
            IF Successful hospital
                IF Confirm continue to update blood donation record
                    Back to step 2
                ELSE
                    Back to staff control menu
            ELSE
                Display "Query Execution Problem"
                Back to StaffControlUpdate()
        ELSE
            Display "Query Execution Problem"
            Back to StaffControlUpdate()
```

ELSE IF (StaffUpdate == 'M' || StaffUpdate == 'm')

5.9 Back to Main Menu

ELSE

5.10 Display "Invalid Choice!"

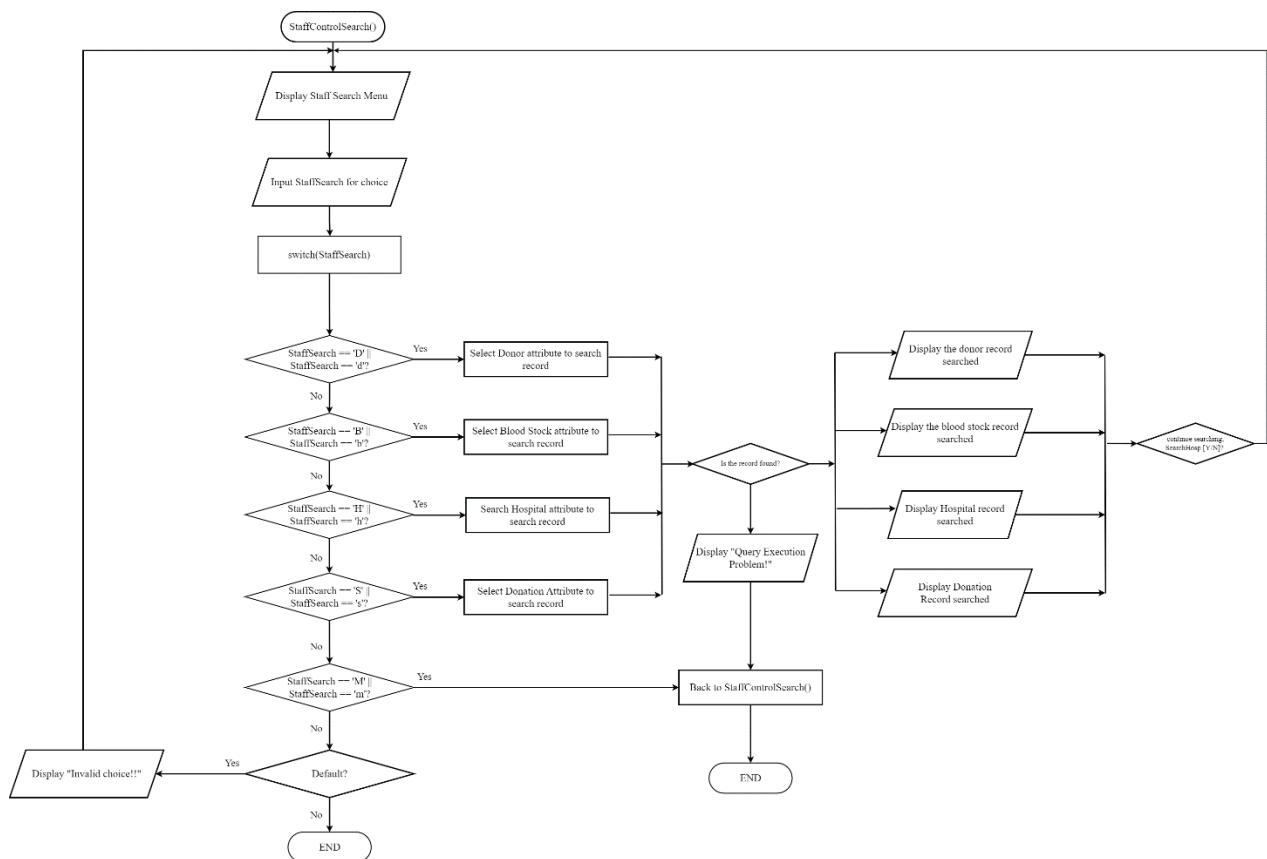
5.11 Back to step 2

END IF

6. END

3.1.12 Staff Control Menu – Search

Flowchart



Pseudocode

1. Start
2. Display Staff control delete menu
3. Input StaffSearch for choice
4. Switch (StaffSearch)
 5. IF (StaffSearch == 'D' || StaffSearch == 'd')
 - 5.1 Select donor attribute to be used to search donor record
 - 5.2 Input based on the attribute selected
 - 5.3 IF record found,
 - 5.3.1 Display record
 - IF continue search donor record
 - Back to Step 5

```
        ELSE
            Back to Step 2
    ELSE
        5.3.2 Display “Query Execution Problem”
        5.3.3 Back to StaffControlSearch()

ELSE IF (StaffSearch == ‘H’ || StaffSearch == ‘h’)
    5.4 Select hospital attribute to be used to search hospital record
    5.5 Input based on the attribute selected
    5.6 IF record found,
        5.6.1 Display record
            IF continue search hospital record
                Back to Step 5
            ELSE
                Back to Step 2
        ELSE
            5.6.2 Display “Query Execution Problem”
            5.6.3 Back to StaffControlSearch()

ELSE IF (StaffDelete == ‘B’ || StaffDelete == ‘b’)
    5.7 Select blood stock attribute to be used to search blood stock record
    5.8 Input based on the attribute selected
    5.9 IF record found,
        5.9.1 Display record
            IF continue search blood stock record
                Back to Step 5
            ELSE
                Back to Step 2
        ELSE
            5.9.2 Display “Query Execution Problem”
            5.9.3 Back to StaffControlSearch()
```

ELSE IF (StaffDelete == 'S' || StaffDelete == 's')

5.10 Select blood donation attribute to be used to search blood donation record

5.11 Input based on the attribute selected

5.12 IF record found,

5.12.1 Display record

IF continue search blood donation record

Back to Step 5

ELSE

Back to Step 2

ELSE

5.12.2 Display “Query Execution Problem”

5.12.3 Back to StaffControlSearch()

ELSE IF (StaffDelete == 'M' || StaffDelete == 'm')

5.13 Back to Main Menu

ELSE

5.14 Display “Invalid Choice!”

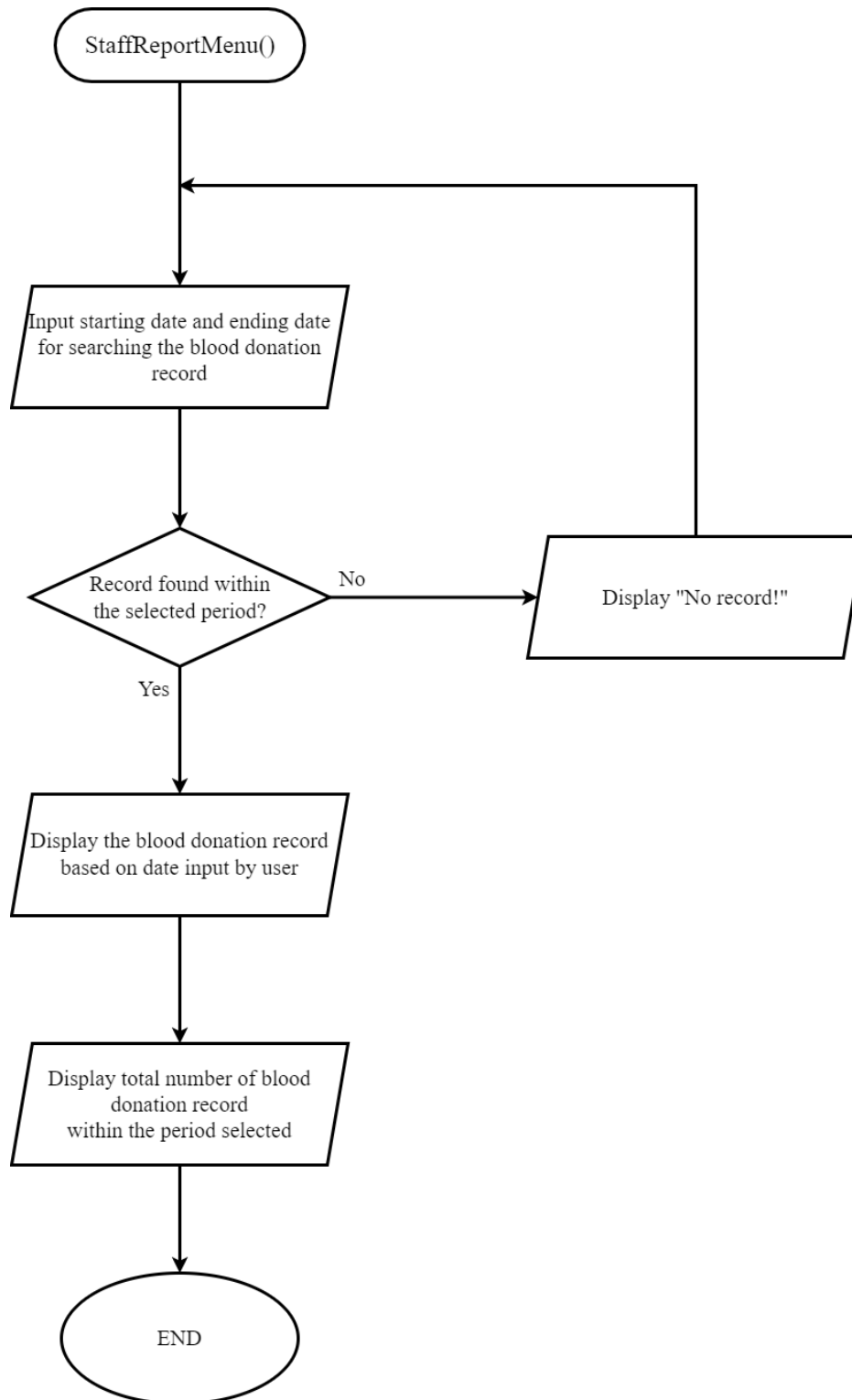
5.15 Back to step 2

END IF

6. END

3.1.13 Staff Report Generation

Flowchart

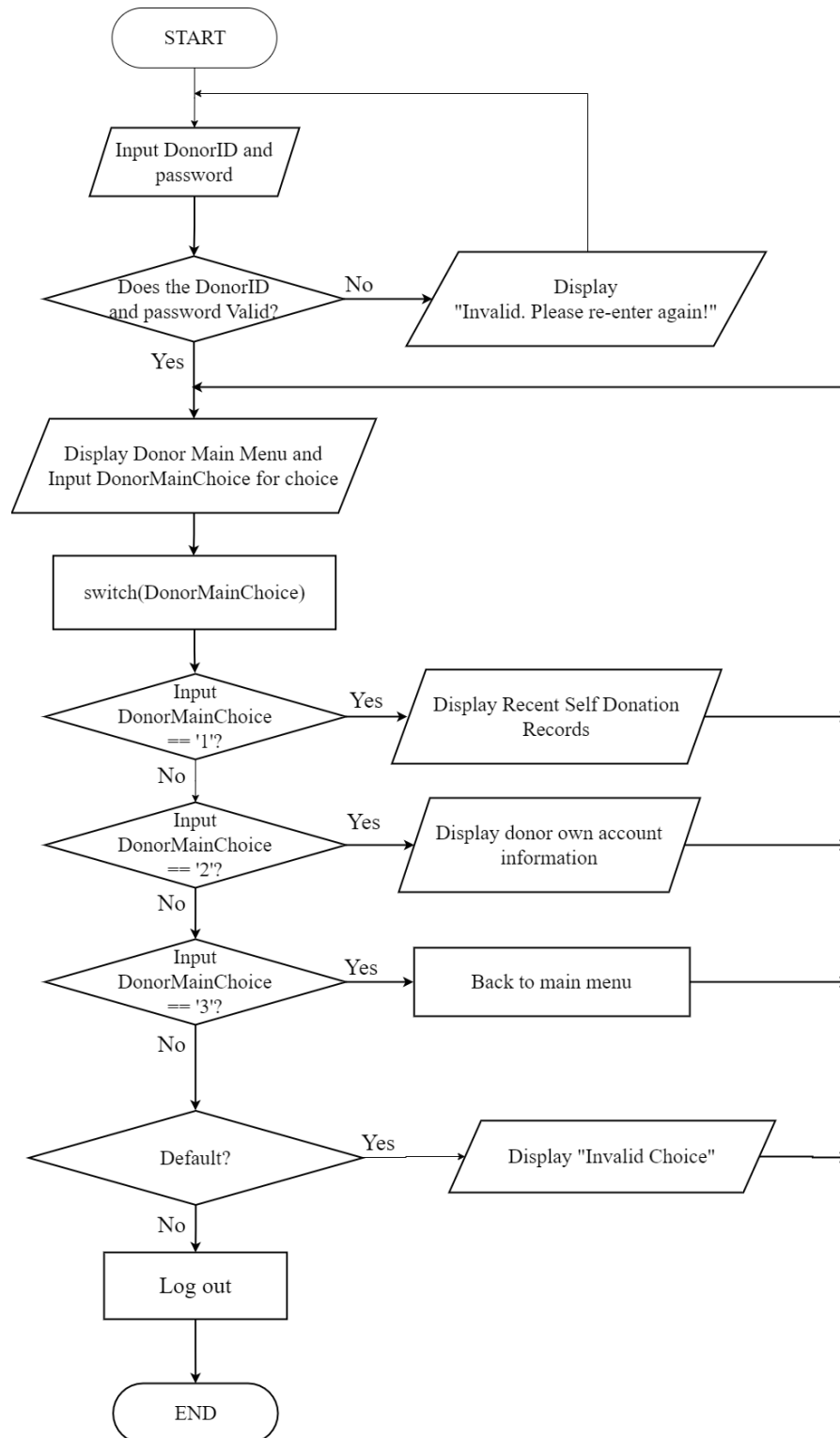


Pseudocode

1. START
2. Input starting date and ending date to search the blood donation
3. IF the blood donation record is found
 - 3.1 Display all blood donation record within the period
 - 3.2 Display total number of blood donation record within the selected period
 - 3.3 Proceed to step 4ELSE Display “No record!”
 - 3.4 Back to step 2END IF
4. END

3.1.14 Donor Main Menu

Flowchart



Pseudocode

1. START
2. Input DonorID and Password
3. IF the DonorID and Password are valid,
 - 3.1 Proceed to step 4ELSE Display “Invalid. Please re-enter again!”
 - 3.2 Back to step 2
4. Display Donor Main Menu and input DonorMainChoice for choice
5. IF the input DonorMainChoice is valid,
 - 5.1 Proceed to step 6ELSE Display “Invalid choice!”
 - 5.2 Back to step 4
6. Switch(DonorMainChoice)
7. IF the input DonorMainChoice == ‘1’,
 - 7.1 Display Self Recent Donation Records
 - 7.2 Back to step 4ELSE IF the input DonorMainChoice == ‘2’,
 - 7.3 Display Donor own account information
 - 7.4 Back to step 4ELSE IF the input DonorMainChoice == ‘3’,
 - 7.5 Back to Main MenuELSE Proceed to step 8
8. Log out
9. END

3.3 Business Rule

1. A blood stock only can add into a blood donation while a blood donation record can be added to a blood stock only.
2. A donor can have more than one donor record and a hospital can store multiple data about donor record.
3. A staff manages one or more blood stock while a blood stock can be only managed by a staff.
4. A staff has more than one staff record and a staff record includes a staff only.
5. A hospital can have more than one staff record but a staff record can be included in a hospital only.
6. A staff is managed by an admin and an admin can manage multiple staffs.
7. A blood donation includes only a donor record but a donor record can be included in more than one blood donation.
8. A blood donation can include only one blood stock but a blood stock can be included in multiple blood donations.

3.4 Data Dictionary

Table 3.4.1 Data Dictionary for Table HOSPITAL

HOSPITAL						
Attribute Name	Description	Data type	Field length	Format	PK or FK	FK Referenced Table
Hospital_ID	Hospital Code	CHAR	6	HS####	PK	
Hospital_Name	Hospital Name	VARCHAR	20			
Hospital_Street	Hospital Street	VARCHAR	40			
Hospital_State	Hospital State	VARCHAR	20			
Hospital_City	Hospital City	VARCHAR	15			
Availability	Availability	VARCHAR	11			

Table 3.4.2 Data Dictionary for Table STAFF

STAFF						
Attribute Name	Description	Data type	Field length	Format	PK or FK	FK Referenced Table
Staff_ID	Staff Code	CHAR	6	ST####	PK	
Staff_Name	Staff Name	VARCHAR	20			
Staff_Gender	Staff Gender	CHAR	1			
Staff_Age	Staff Age	NUMBER	2			
Staff_Address	Staff Address	VARCHAR	50			
Staff_TelNo	Staff Phone Number	VARCHAR	14			
Staff_Position	Staff Position	VARCHAR	20			
Staff_Password	Staff Password	VARCHAR	40			
Admin_ID	Admin Code	CHAR	6	ST####	FK	STAFF
Hospital_ID	Hospital Code	CHAR	6	HS####	FK	HOSPITAL
Active_Status	Active Status	VARCHAR	8			

Table 3.4.3 Data Dictionary for Table STAFF_RECORD

STAFF_RECORD						
Attribute Name	Description	Data type	Field length	Format	PK or FK	FK Referenced Table
Staff_ID	Staff Code	CHAR	6	ST####	PK, FK	STAFF
Hospital_ID	Hospital Code	CHAR	6	HS####	PK, FK	HOSPITAL

Table 3.4.4 Data Dictionary for Table DONOR

DONOR						
Attribute Name	Description	Data type	Field length	Format	PK or FK	FK Referenced Table
Donor_ID	Donor Code	CHAR	6	DN####	PK	
Donor_Name	Donor Name	VARCHAR	20			
Donor_Age	Donor Age	NUMBER	2			
Donor_Gender	Donor Gender	CHAR	1			
Donor_DOB	Donor Date of Birth	DATE		DD-MM-YY		
Donor_Address	Donor Address	VARCHAR	50			
Donor_BloodType	Donor Blood Type	CHAR	2			
Donor_Weight	Donor Weight	FLOAT	6			
Donor_Height	Donor Height	FLOAT	6			
Donor_TelNo	Donor Phone Number	VARCHAR	14			
Donor_Password	Donor Password	VARCHAR	40			
Active_Status	Active Status	VARCHAR	8			

Table 3.4.5 Data Dictionary for Table DONOR_RECORD

DONOR_RECORD						
Attribute Name	Description	Data type	Field length	Format	PK or FK	FK Referenced Table
Record_ID	Record Code	CHAR	6	RC####	PK	
Donor_ID	Donor Code	CHAR	6	DN####	PK, FK	DONOR
Hospital_ID	Hospital Code	CHAR	6	HS####	PK, FK	HOSPITAL

Table 3.4.6 Data Dictionary for Table BLOOD_DONATION

BLOOD_DONATION						
Attribute Name	Description	Data type	Field length	Format	PK or FK	FK Referenced Table
Donation_ID	Donation Code	CHAR	6	DT####	PK	
Donation_Date	Donation Name	DATE		DD-MM-YY		
Donation_Volume	Donation Volume	NUMBER	5			
Record_ID	Record Code	CHAR	6	RC####	PK	DONOR_RECORD
Status	Status (Completed / Unavailable)	VARCHAR	11			

Table 3.4.7 Data Dictionary for Table BLOOD_STOCK

BLOOD_STOCK						
Attribute Name	Description	Data type	Field length	Format	PK or FK	FK Referenced Table
Stock_ID	Blood Stock Code	CHAR	6	BL####	PK	
Stock_Category	Type of blood stock	CHAR	1			
Stock_Volume	Blood Stock Volume	INT				
Stock_Quantity	Blood Stock Quantity	INT				
Total_Stock_Volume	Total Stock Volume	INT				
Staff_ID	Staff Code	CHAR	6	ST####	FK	STAFF
Donation_ID	Donation Code	CHAR	6	DT####	FK	BLOOD_DONATION
Availability	Availability	VARCHAR	11			

3.5 Interface Design

In this system, the interface is designed in a command-line based. Command-line interface is a text-based interface where users have interaction with the system by typing some text as commands on specific lines following the visual prompt from the computer. It's also known as console user interfaces.

Using this command-line interface (CLI), user may issue a series of commands that are received by the system and will be executed later. command line interface (CLI) is a text-based user interface (UI) used to view and manage computer files.

Computer users especially administrators may fully utilize this interface as it is different from the graphical user interface (GUI) that is being used in present. However, its working mechanism is not user-friendly although it's easy to use as users have to press "Enter" button on keyboard and also wait for the system to response.

In this system, CLI will be used as the console to control the operation of system based on different user views such as admin, staffs and donors. Each of them has different type of access to use the system and hence the functionality for each user will be different too.

3.5.1 Main Menu

```
-----  
Welcome To Blood Donation Management System  
-----  
  
[1] Login  
[2] Register  
[3] Exit  
  
Enter your option:
```

3.5.2 Login

```
-----  
LOGIN  
-----  
  
[1] Login As Admin  
[2] Login As Staff  
[3] Login As Donor  
[4] Forget Password  
[5] Back to Main Menu  
  
Enter your choice (Number 1 - 5 only):
```

3.5.3 Admin Login

```
-----  
  LOGIN AS ADMIN  
-----  
Enter Admin ID:  
Enter Password:
```

3.5.4 Staff Login

```
-----  
  LOGIN AS STAFF  
-----  
Enter Staff ID:  
Enter Password:
```

3.5.5 Donor Login

```
-----  
  LOGIN AS DONOR  
-----  
Enter Donor ID:  
Enter Password:
```

3.5.6 User Account Registration Menu

```
-----  
ACCOUNT REGISTRATION  
-----  
Please choose which role do you want to register?"  
[1] As Admin / Staff  
[2] As Donor  
[3] Main Menu  
  
Your choice (1 - 3):
```

3.5.7 Admin/Staff Registration

```
-----  
NEW ADMIN/STAFF REGISTRATION  
-----  
Please fill in the information below:  
  
Name:  
Gender:  
Address:  
Age:  
Telephone Number:  
Position:  
Admin ID:  
  
-----  
Now, please set your Staff ID and password:  
  
Staff ID (Format: ST****):  
Password:  
Re-enter password:  
  
You have been registered. Press Enter to Continue...
```

3.5.8 Donor Registration

NEW DONOR REGISTRATION

Please fill in the information below:

Name:

Gender:

Date of Birth (YYYY MM DD):

Address:

Blood Type:

Height:

Weight:

Telephone Number:

Now please set your Donor ID and password:

Donor ID (Format: DN****):

Password:

Re-enter password:

You have been registered. Press Enter to Continue...

3.5.9 Admin Main Menu

```
-----  
ADMIN MAIN MENU  
-----  
  
Welcome, admin!  
  
[1] Administration Control  
[2] Staff Records  
[3] Report Generation  
[4] Back to Main Menu  
  
Please enter your choice:
```

3.5.10 Admin Control Menu

```
-----  
ADMIN CONTROL MENU  
-----  
  
Welcome, admin! What would you like to do with staff?  
  
[A] Add Staff Record  
[B] Delete Staff Record  
[C] Update Staff Record  
[D] Search Staff Record  
[M] Main Menu  
  
Please enter your choice:
```

3.5.11 Admin Control Menu – Add Staff Record

```
-----  
ADMIN ADD - STAFF  
-----  
  
Enter new records:  
Staff ID      :  
Name          :  
Gender        :  
Age           :  
Address       :  
Telephone Number :  
Position      :  
Staff Password :  
Hospital ID   :  
  
Done.  
A staff is successfully added in database.  
Do you want add another record? (Y/N):
```

3.5.12 Admin Control Menu – Delete Staff Record

```
-----  
ADMIN DELETE - STAFF  
-----
```

```
Enter staff ID to search:
```

```
Here's the record found:
```

```
Staff ID      :  
Name          :  
Gender        :  
Age           :  
Address       :  
Telephone Number :  
Position      :  
Hospital ID   :
```

```
Are you confirm to delete this record? [Y/N]:
```

```
Successfully delete a staff!
```

```
Do you want to continue deleting? [Y/N]:
```


3.5.13 Admin Control Menu – Search Staff Record

```
-----  
ADMIN SEARCH - STAFF  
-----  
Enter staff ID to search:  
  
Here's the record found:  
Staff ID      :  
Name          :  
Gender        :  
Age           :  
Address       :  
Telephone Number :  
Position      :  
Password      :  
Hospital ID   :  
  
Do you want to continue searching? [Y/N]:
```

3.5.14 Admin Control Menu – Update Staff Record

```
-----  
ADMIN UPDATE - STAFF  
-----  
Enter staff ID to search:  
  
Here's the record found:  
Staff ID      :  
Name          :  
Gender        :  
Age           :  
Address       :  
Telephone Number :  
Position      :  
Hospital ID   :  
  
-----  
Updation of information  
  
Enter information that you want to update  
(Type the same details back if you don't want to  
update for that part)  
  
Enter information that you want to update:  
[1] Name  
[2] Gender  
[3] Age  
[4] Address  
[5] Phone  
[6] Position  
[7] Password  
[8] Hospital ID  
  
Your choice >>  
  
Successfully Updated!  
Do you want to continue update record? [Y/N]:
```

3.5.15 Admin Display Staff Record

```
-----  
ADMIN DISPLAY - STAFF  
-----  
Here is the record of staff:  
  
-----  
| Staff ID | Name | Gender | Age | Address | Telephone Number | Position | Hospital ID |  
-----  
  
Press any key to continue....
```

3.5.16 Admin Report Generation

```

-----
REPORT GENERATION FOR ADMIN
-----

The default number of active staffs in the system:
Which attribute do you want to generate as report?

[1] Gender
[2] Position
[3] Age
[4] Back to Main Menu

Your choice:

```

3.5.17 Admin Report Generation By Age

```

The number of active staff based on age:
20-29 :
30-39 :
40-49 :
50-59 :

*****
|          Graph (AGE VS NUMBER OF STAFF)          |
*****
Gender          Number of Staff
  /  \
20-29  |****  4
30-39  |*****  7
40-49  |***   3
50-59  |
          ----->

Do you want to continue viewing report? [Y/N]:

```

3.5.18 Admin Report Generation By Gender

```

The number of active staff based on Gender:
Male: 4
Female: 7

*****
|          Graph (GENDER VS NUMBER OF STAFF)          |
*****

  Gender  Number of Staff
      /  \
Male    |**** 4
Female  |***** 7
      ----->

Do you want to continue viewing report? [Y/N]:

```

3.5.19 Admin Report Generation By Position

```

The number of active staff based on position:
Doctor : 4
Nurse  : 3
Staff  : 7

*****
|          Graph (POSITION VS NUMBER OF STAFF)          |
*****

  Position  Number of Staff
      /  \
Doctor    |**** 4
Nurse     |***** 7
Staff     |*** 3
      ----->

Do you want to continue viewing report? [Y/N]:

```

3.5.20 Staff Main Menu

```
-----  
  STAFF MAIN MENU  
-----  
  
Welcome, staff!  
  
[1] Control Menu  
[2] Recent Donation Record  
[3] Report Generation  
[4] Account Information  
[5] Exit  
  
Your choice (1 - 5):
```

3.5.21 Staff Control Menu

```
-----  
  STAFF CONTROL MENU  
-----  
  
Welcome, staff! What would you like to do?  
  
[A] Add Record  
[B] Delete Record  
[C] Update Record  
[D] Search Record  
[F] Back to Main Menu  
  
Please enter your choice:
```

3.5.22 Staff Control Menu – Add Record Menu

```

-----
ADD RECORD
-----
Which record do you want to ADD?

[D] Add Record of Donors
[H] Add Record of Hospital
[B] Add Record of Blood Stock
[S] Add Record of Donation
[M] Main Menu

Please enter your choice:

```

3.5.23 Staff Control Menu – Add Donor Record

```

-----
ADD RECORD - DONORS
-----
Enter new records:

Donor ID(Format: DN****):
Name :
Gender :
Date of Birth (YYYY MM DD): :
Address :
Blood Type :
Weight :
Height :
Telephone Number :
Password :

Donor is successful added in database.
Do you want to continue adding records? [Y/N]:

```

3.5.24 Staff Control Menu – Add Hospital Record

```

-----
  ADD RECORD - HOSPITAL
-----
Enter new records:

Hospital ID :
Name :
Street :
City :
State :

Hospital is successful added in database.
Do you want to continue adding records? [Y/N]:

```

3.5.25 Staff Control Menu – Add Donation Record

```

-----
  ADD RECORD - DONATION
-----
Enter new records:

Donation ID           :
Donation Date         :
Donor ID              :
Donation Volume       :
Blood Type(Blood No.) :
Staff-in-charge(Staff ID) :
Hospital(Hospital ID) :

Successfully added!

Do you want to continue adding record? [Y/N]:

```


3.5.26 Staff Control Menu – Add Blood Stock Record

```
-----  
  ADD RECORD - BLOOD STOCK  
-----  
Enter new records:  
  
Blood Stock ID (Format: BL****):  
Blood Stock Category:  
Blood Stock Volume:  
Blood Stock Quantity:  
Staff ID:  
Quantity ID:  
  
A blood stock is successful added in database.  
Do you want to continue adding records? [Y/N]:
```

3.5.27 Staff Control Menu – Delete Record Menu

```
-----  
  DELETE RECORD  
-----  
Which record do you want to DELETE?  
  
[D] Delete Record of Donors  
[H] Delete Record of Hospital  
[B] Delete Record of Blood Stock  
[S] Delete Record of Donation  
[M] Main Menu  
  
Please enter your choice:
```

3.5.28 Staff Control Menu – Delete Donor Record

```
-----  
DELETE RECORD - DONORS  
-----  
Delete the record:  
  
Enter Donor ID to search:  
  
Here is the record found:  
  
Donor ID      :  
Name          :  
Age           :  
Gender        :  
Date of Birth :  
Address       :  
Blood Type    :  
Weight        :  
Height        :  
Telephone Number :  
  
Are you confirm to delete this record? [Y/N]:  
  
Successfully delete a donor!  
  
Do you want to continue deleting record? [Y/N]:
```

3.5.29 Staff Control Menu – Delete Hospital Record

```

-----
DELETE RECORD - HOSPITAL
-----
Enter Hospital ID to search:

Here is the record found:

Hospital ID :
Name       :
Street     :
City       :
State      :

Are you confirm to delete this record? [Y/N]:

Successfully delete a hospital!

Do you want to continue deleting record?[Y/N]

```

3.5.30 Staff Control Menu – Delete Donation Record

```

-----
DELETE RECORD - DONATION
-----
Enter Donation ID to search:

Here is the record found:

Donation ID      :
Date             :
Volume           :
Record ID        :
Hospital ID      :
Donor ID         :

Are you confirm to delete this record? [Y/N]:

Successfully delete a blood donation record!

Do you want to continue deleting record? [Y/N]:

```

3.5.31 Staff Control Menu – Delete Blood Stock Record

```
-----  
DELETE RECORD - BLOOD STOCK  
-----  
Enter Blood Stock ID to search :  
  
Here is the record found:  
  
Stock ID:  
Category:  
Volume:  
Quantity:  
Total Volume:  
Staff ID:  
Donation ID:  
  
Are you confirm to delete this record? [Y/N]:  
  
Successfully delete a blood stock!  
  
Do you want to continue deleting record?[Y/N]:
```

3.5.32 Staff Control Menu – Update Record Menu

```
-----  
UPDATE RECORD  
-----  
Which record do you want to UPDATE?  
  
[D] Update Record of Donors  
[H] Update Record of Hospital  
[B] Update Record of Blood Stock  
[S] Update Record of Donation  
[M] Main Menu  
  
Please enter your choice:
```

3.5.33 Staff Control Menu – Update Donor Record

```
-----
UPDATE RECORD - DONOR
-----
Enter Donor ID to search (Format:DN****) :

Here is the record found:

Donor ID      :
Name          :
Age           :
Gender        :
Date of Birth :
Address       :
Blood Type    :
Weight        :
Height        :
Telephone Number :

-----
Updation of information

Enter information that you want to update:

[1] Name
[2] Gender
[3] Date of Birth
[4] Address
[5] Blood Type
[6] Height
[7] Weight
[8] Phone
[9] Password
Your choice >>

Successfully updated! Do you want to continue update
record? [Y/N]:
```

3.5.34 Staff Control Menu – Update Hospital Record

```
-----  
UPDATE RECORD - HOSPITAL  
-----  
Enter Hospital ID to search(Format: HS****):  
  
Here is the record found:  
  
Hospital ID :  
Name       :  
Street     :  
City       :  
State      :  
  
-----  
Updation of information  
  
Enter information that you want to update:  
[1] Name  
[2] Street  
[3] City  
[4] State  
  
Your choice >>  
  
Successfully updated! Do you want to continue update  
record? [Y/N]:
```

3.5.35 Staff Control Menu – Update Donation Record

```
-----  
UPDATE RECORD - DONATION  
-----  
Enter Donation ID to search (Format: DT***):  
  
Here is the record found:  
  
Record ID      :  
Donor ID       :  
Donation ID    :  
Donation Date  :  
Donation Volume :  
Hospital ID    :  
  
-----  
Updation of information  
  
Enter information that you want to update  
Enter information that you want to update  
  
[1] Record ID  
[2] Donor ID  
[3] Donation Date  
[4] Donation Volume  
[5] Hospital ID  
  
Your choice >>  
  
Successfully updated! Do you want to continue update  
record? [Y/N]:
```

3.5.36 Staff Control Menu – Update Blood Stock Record

```
-----  
UPDATE RECORD - BLOOD STOCK  
-----  
Enter Blood Stock ID to search (Format: BL****) :  
  
Here is the record found:  
  
Stock ID           :  
Category           :  
Volume             :  
Quantity           :  
Total Volume       :  
Staff ID           :  
Donation ID        :  
  
-----  
Updation of information  
  
Enter information that you want to update:  
  
[1] Category  
[2] Volume  
[3] Quantity  
[4] Staff ID  
[5] Donation ID  
  
Your choice >>  
  
Successfully updated! Do you want to continue update  
record? [Y/N]:
```


3.5.37 Staff Control Menu – Search Record Menu

```
-----  
SEARCH RECORD  
-----  
Which record do you want to SEARCH?  
  
[D] Search Record of Donors  
[H] Search Record of Hospital  
[B] Search Record of Blood Stock  
[S] Search Record of Donation  
[M] Main Menu  
  
Please enter your choice:
```

3.5.38 Staff Control Menu – Search Donor Record

```
-----  
SEARCH RECORD - DONOR  
-----  
Please select the attribute you want to search:  
  
[1] Donor ID  
[2] Name  
[3] Gender  
[4] Age  
[5] Date of Birth  
[6] Address  
[7] Blood Type  
[8] Height  
[9] Weight  
[10] Phone  
[11] Back to Main Menu  
  
Your Choice >>
```

```
Your Choice >> 1  
  
Enter Donor ID to search:  
  
Here is the record found:  
  
Donor ID      :  
Name         :  
Age          :  
Gender       :  
Date of Birth :  
Address      :  
Blood Type   :  
Weight       :  
Height       :  
Phone        :  
Password     :  
  
Do you want to search other donor? [Y/N]:
```

3.5.39 Staff Control Menu – Search Hospital Record

```
-----  
SEARCH RECORD - HOSPITAL  
-----  
Please select the attribute you want to search:  
[1] Hospital ID  
[2] Name  
[3] Street  
[4] City  
[5] State  
[6] Back to Main Menu  
  
Your Choice >>
```

```
Your Choice >> 1  
  
Enter Hospital ID to search:  
  
Here is the record found:  
  
Hospital ID :  
Name       :  
Street      :  
City        :  
State       :  
  
Do you want to search other hospital? [Y/N]:
```

3.5.40 Staff Control Menu – Search Donation Record

```
-----  
SEARCH RECORD - DONATION  
-----  
Please select the attribute you want to search:  
[1] Record ID:  
[2] Donation ID  
[3] Donor ID  
[4] Donation Date  
[5] Donation Volume  
[6] Hospital ID  
[7] Back to Main Menu  
  
Your Choice >>
```

```
Your Choice >> 1  
  
Enter Donation ID to search:  
  
Here is the record found:  
  
Record ID:  
Donation ID:  
Donor ID:  
Donation Date:  
Donation Volume:  
Hospital ID:  
  
Do you want to search other blood donation record? [Y/N]:
```

3.5.41 Staff Control Menu – Search Blood Stock Record

```
-----  
SEARCH RECORD - BLOOD STOCK  
-----
```

Please select the attribute you want to search:

- [1] Stock ID
- [2] Category
- [3] Volume
- [4] Quantity
- [5] Total Volume
- [6] Staff ID
- [7] Donation ID
- [8] Back to Main Menu

Your choice >>

Your choice >> 1

Enter Blood Stock ID to search:

Here is the record found:

Stock ID:

Category:

Volume:

Quantity:

Total Volume:

Staff ID:

Donation ID:

Do you want to continue searching [Y/N]:

3.5.42 Staff Control Menu – Donation Record Menu

```

-----
BLOOD DONATION RECORD
-----
Here is the blood donation record:

-----
Record ID  Donation Date  Donor ID  Donor Name  Donation Volume  Blood Type  Hospital
-----

Press any key to continue....

```

3.5.43 Staff Control Menu – Report Generation

```

=====
STAFF REPORT
=====

=====
BLOOD DONATION REPORT
=====

-----
REPORT SELECTION QUERY
-----

Please select the period for blood donation record:

If want to choose same year, just type same value for both starting and ending year;
If want to choose same day, just type same value for both starting and ending day;
If want to choose same month, just type same value for both starting and ending month;

Select starting year (YYYY):
Select starting month (MM):
Select starting day (DD):
Select ending year (YYYY):
Select ending month (MM):
Select ending day (DD):

Here's the record found:

-----
| Record | Donor Name | Donation Date | Volume | Hospital |
-----

```

3.5.44 Staff Account Information

```
=====
ACCOUNT INFORMATION
=====

Staff ID:
Name:
Gender:
Age:
Address:
Phone:
Position:
Hospital ID:
```

3.5.45 Donor Main Menu

```
-----
DONOR MAIN MENU
-----

Welcome, donor!

[1] Self Recent Donation
[2] Account Information
[3] Back to main menu

Please enter your choice (1 - 3):
```

3.5.46 Donor Display Recent Donation Record

```
-----  
RECENT DONATION RECORD  
-----  
  
Donor ID:  
Name:  
  
Here is the sorted record of yours as a donor:  
  
-----  
| Record ID | Name | Date | Volume | Hospital |  
-----  
  
Press any key to continue....
```

3.5.47 Donor Own Account Information

```
=====
```

ACCOUNT INFORMATION
Donor ID:
Name:
Gender:
Age:
Date of Birth:
Address:
Blood Type:
Height:
Weight:
Phone:

```
=====
```


3.5.48 Admin/Staff Password Recovery

```
-----  
PASSWORD RECOVERY  
-----  
  
Are you a?  
[1] Admin / Staff  
[2] Donor  
  
Your choice: 1  
  
Please Insert your staff ID:  
  
Password:  
  
Successfully Updated!
```

3.5.49 Donor Password Recovery

```
-----  
PASSWORD RECOVERY  
-----  
  
Are you a?  
[1] Admin / Staff  
[2] Donor  
  
Your choice: 2  
  
Please Insert your Donor ID:  
  
Password:  
  
Successfully Updated!
```

CHAPTER 4 IMPLEMENTATION

4.1 Introduction

System implementation is the phase in which the initial design is translated into software code and logic procedures. During this phase, we test the system to ensure that development has been successful and that user needs are being met. The implementation phase includes building a working system and experimentation process. It is also the stage at which design decisions are reflected in software code and logic procedures. This phase tests the software to ensure that the integrated software meets the requirements criteria. In some processes, commonly known as test-driven development, tests can be developed just before implementation to guide the correctness of the implementation.

4.2 Data Manipulation Language (DML)

4.2.1 Listing Table Rows

The SELECT command is used to list the contents of a table. Below is one of the SELECT commands that is being implemented.

SQL – Unary Operation

```
string search_query = "SELECT Staff_ID, Staff_Name, Staff_Gender,
Staff_Age, Staff_Address, Staff_Telno, Staff_Position FROM staff
WHERE Staff_ID = '" + id + "' AND Active_Status = 'Active'";
const char* q = search_query.c_str();
qstate = mysql_query(conn, q);
```

SQL – Aggregation and Grouping Operation

```
string countGender_query = "Select SUM(case when Staff_Gender =
'M' then 1 else 0 end), SUM(case when Staff_Gender = 'F' then 1
else 0 end) FROM staff WHERE Active_Status = 'Active'";
const char* cu1 = countGender_query.c_str();
qstate = mysql_query(conn, cu1);
```

SQL – Join Operation

```
string viewDonorRecordList = "SELECT s.Record_ID, t.Donation_Date,
d.Donor_ID, d.Donor_Name, t.Donation_Volume, d.Donor_BloodType,
h.Hospital_Name FROM donor d, blood_donation t, donor_record s ,
hospital h WHERE d.Donor_ID = s.Donor_ID AND t.Record_ID =
```

```
s.Record_ID AND h.Hospital_ID = s.Hospital_ID ORDER BY
s.Record_ID";
const char* vtr = viewDonorRecordList.c_str();
qstate = mysql_query(conn, vtr);
```

4.2.2 Inserting Table Rows

Insert command is used to enter data into a table. Below is one of the INSERT commands that is being implemented.

```
string insert_query = "INSERT INTO donor (Donor_ID, Donor_Name,
Donor_Gender, Donor_Age, Donor_DOB, Donor_Address, Donor_BloodType,
Donor_Height, Donor_Weight, Donor_TelNo, Donor_Password, Active_Status)
VALUES ('" + Donor_ID + "', '" + Donor_Name + "', '" + Donor_Gender +
"', '" + to_string(Donor_Age) + "', '" + Donor_DOB + "', '" +
Donor_Address + "', '" + Donor_BloodType + "', '" +
to_string(Donor_Height) + "', '" + to_string(Donor_Weight) + "', '" +
Donor_TelNo + "', sha1('" + Donor_Password + "'), '" + Active_Status +
"')";
const char* q = insert_query.c_str();
qstate = mysql_query(conn, q);
if (!qstate)
{
    cout << endl << "Donor is successful added in database." <<
endl;
}
else
{
    cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
}
```

4.2.3 Updating Table Rows

The update command is used to change the data of a table. Below is one of the UPDATE commands that is being implemented.

```
string update_query = "UPDATE blood_stock SET Stock_Category = '"
+ Stock_Category + "' WHERE Stock_ID = '" + Stock_ID + "'";
const char* q = update_query.c_str();
qstate = mysql_query(conn, q);
```

4.2.4 Deleting Table Rows

Delete command is used to delete data from a table. Below is one of the DELETE commands that is being implemented.

```

string delete_query = "DELETE FROM blood_stock where Stock_ID =
'" + Stock_ID + "'";
const char* q = delete_query.c_str();
qstate = mysql_query(conn, q);

```

4.3 Programming Language

Below are the specifications of the project file:

Build Managing tool: Microsoft Visual Studio 2019 and 2022

Language : C++ & MySQL

4.3.1 Database Connection

Below are the implementation and connection of the database:

```

int qstate;
MYSQL* conn;
MYSQL_ROW row;
MYSQL_RES* res;

//class for database connection
class db_response
{
public:
    static void ConnectionFunction()
    {
        //check whether if database connected
        conn = mysql_init(0);
        if (conn)
        {
            cout << "Database Connected!" << endl;
            system("cls");
        }
        else
        {
            cout << "Failed To Connect!" << mysql_errno(conn) << endl;
        }

        //check whether if database connected
        conn = mysql_real_connect(conn, "localhost", "root", "",
            "db_workshop1_sem2", 3306, NULL, 0);
        if (conn)
        {
            cout << "Database db_workshop1_sem2 Connected To MySql!" << endl;
            //getch();
        }
        else
    }
}

```

```

    {
        cout << "Failed To Connect!" << mysql_errno(conn) << endl;
    }
};

```

```
Database db_workshop1_sem2 Connected To MySQL!
```

```
Date: 6/1/2023
```

```
Time: 2:42:00 pm
```

```
-----
Welcome to Blood Donation Management System
-----
```

```
[1] Login
[2] Register
[3] Exit
```

```
Enter your option:
```

4.4 Selection

4.4.1 Selection (Switch Case)

User will input choice and based on the input will lead to different function modules. The following is the example of staff adding module to choose which record does the staff need to add:

```

switch (StaffAdd) {
    case 'H':
    case 'h':
        AddHospitalMenu();
        break;

    case 'D':
    case 'd':
        AddDonorMenu();
        break;

    case 'B':

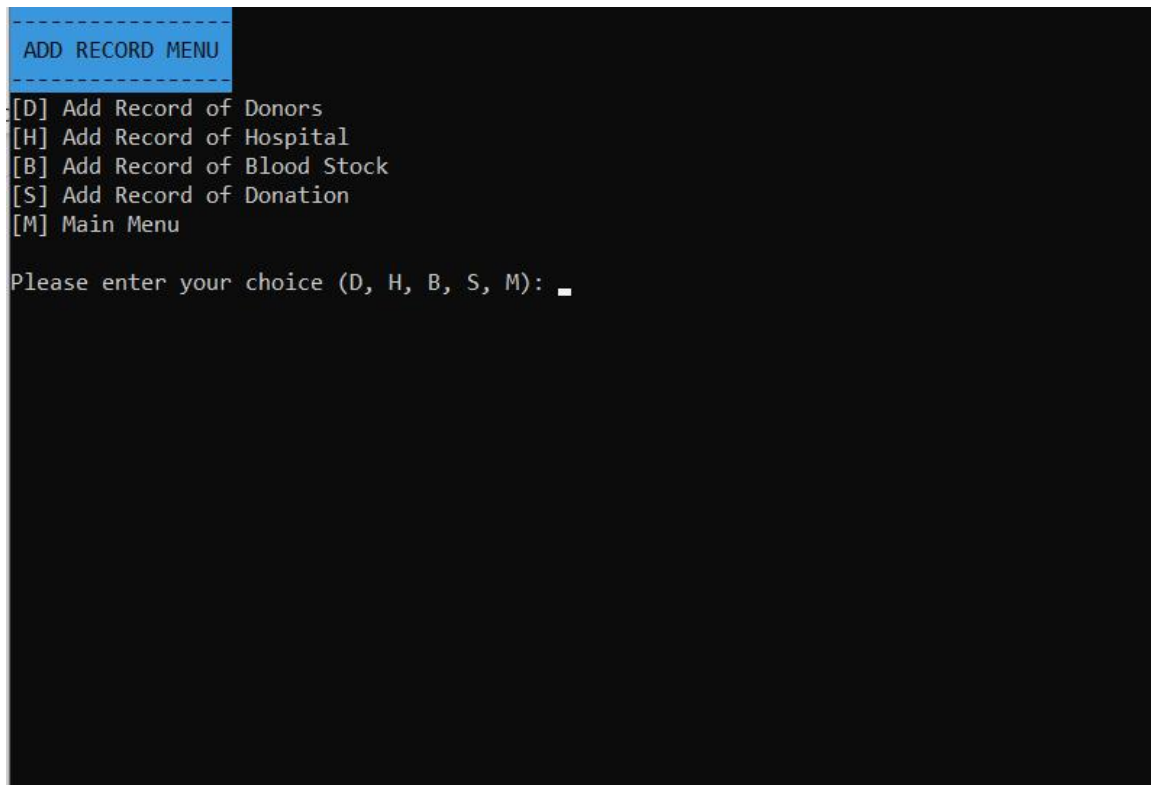
```

```
case 'b':
    AddBloodMenu();
    break;

case 'S':
case 's':
    AddDonationMenu();
    break;

case 'M':
case 'm':
    StaffControlMain();
    break;

default:
    cout << "Invalid choice!" << endl;
    system("pause");
    StaffControlAdd();
    break;
}
```



```
-----
ADD RECORD MENU
-----
[D] Add Record of Donors
[H] Add Record of Hospital
[B] Add Record of Blood Stock
[S] Add Record of Donation
[M] Main Menu

Please enter your choice (D, H, B, S, M): _
```

4.4.2 Selection (If-Else)

User will input choice and based on the input will lead to different function modules. The following is the example of when a user forgets his or her password so the system will check based on the role and the user id that input by user, if the user exist then will let the user update his password then try login again.

```

if (recover == '1')
{
    string Staff_Password;
    system("cls");
    cout << "-----" << endl;
    cout << " ADMIN/STAFF " << endl;
    cout << "-----" << endl;
    cout << "\nPlease insert your staff ID: " << endl;
    cin >> Staff_ID;

    string search_query = "SELECT Staff_ID FROM staff WHERE Staff_ID = '" +
Staff_ID + "' AND Active_Status = 'Active'";
    const char* q = search_query.c_str();
    qstate = mysql_query(conn, q);
    if (!qstate)
    {
        res = mysql_store_result(conn);
        while (row = mysql_fetch_row(res))
        {
            Staff_ID = row[0];
        }
        cout << "Staff found!" << endl;
        cout << "Set your new password: ";
        char ch;
        while ((ch = _getch()) != 13)
        {
            Staff_Password += ch;
            cout << "*";
        }

        string update_query = "UPDATE staff SET Staff_Password = sha1('" +
Staff_Password + "') WHERE Staff_ID = '" + Staff_ID + "'";
        const char* q = update_query.c_str();
        qstate = mysql_query(conn, q);
        cout << "\n Successfully Updated!" << endl;
        system("pause");
        MainLogin();
    }
    else
    {
        cout << "Sorry, no such ID exist! Please try again!" <<
mysql_errno(conn) << endl;
        system("pause");
        MainLogin();
    }
}

```

```

else if (recover == '2')
{
    string Donor_Password;
    system("cls");
    cout << "-----" << endl;
    cout << " DONOR " << endl;
    cout << "-----" << endl;
    cout << "\nPlease insert your donor ID: " << endl;
    cin >> Donor_ID;

    string search_query = "SELECT Donor_ID FROM donor WHERE Donor_ID = '" +
Donor_ID + "' AND Active_Status = 'Active'";
    const char* q = search_query.c_str();
    qstate = mysql_query(conn, q);
    if (!qstate)
    {
        res = mysql_store_result(conn);
        while (row = mysql_fetch_row(res))
        {
            Donor_ID = row[0];
        }
        cout << "Donor found!" << endl;
        cout << "Set your new password: ";
        char ch;
        while ((ch = _getch()) != 13)
        {
            Donor_Password += ch;
            cout << "*";
        }

        string update_query = "UPDATE donor SET Donor_Password = sha1('" +
Donor_Password + "') WHERE Donor_ID = '" + Donor_ID + "'";
        const char* q = update_query.c_str();
        qstate = mysql_query(conn, q);
        cout << "\n Successfully Updated!" << endl;
        system("pause");
        MainLogin();
    }
    else
    {
        cout << "Sorry, no such ID exist! Please try again!" <<
mysql_errno(conn) << endl;
        system("pause");
        MainLogin();
    }
}
else if (recover == 'M' || recover == 'm')
{
    MainLogin();
}
else
{
    cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
    system("pause");
    MainLogin();
}
}

```



```
-----  
PASSWORD RECOVERY  
-----  
  
Are you a ?  
[1] Admin/Staff  
[2] Donor  
[M] Main Menu  
  
Your choice: █
```

```
-----  
ADMIN/STAFF  
-----  
  
Please insert your staff ID: ST0012  
Staff found!  
Set your new password:
```

```
-----  
DONOR  
-----  
  
Please insert your donor ID: DN0010  
Donor found!  
Set your new password: █
```

4.5 Control

4.5.1 Control (For Loop)

```
//Display the graph
cout << "\n" << endl;
cout << background(Light_Red);
cout << "*****" << endl;
cout << "|          Graph (GENDER VS NUMBER OF STAFF)          |" << endl;
cout << "*****" << endl;
cout << background(Black);
cout << endl;
cout << setw(6) << "Gender" << right << " Number of Staff" << endl;
cout << setw(8) << right << "/" << endl;

for (int i = 0; i < 2; i++)
{
    cout << setw(6) << left << arrayGenderType[i] << left << "|";

    for (int j = 0; j < arrayGender[i]; j++)
    {
        cout << "*";
    }
    if (arrayGender[i] != 0) //showing the number of each gender in the
graph
    {
        cout << " " << foreground(Yellow) << arrayGender[i] <<
foreground(White) << endl;
    }
    else
    {
        cout << endl;
    }
}

cout << setw(7) << right;
int* maxNum;
maxNum = max_element(arrayGender, arrayGender + 1);
for (int i = 0; i < *maxNum + 5; i++)
{
    cout << "-";
}
cout << ">" << endl;
```

```

Date: 6/1/2023           Time: 2:52:16 pm

The number of active staff based on gender:
Male = 13
Female = 7

The following graph showing the number of staff based on gender:

*****
|      Graph (GENDER VS NUMBER OF STAFF)      |
*****

Gender Number of Staff
  / \
Male |***** 13
Female|***** 7
----->

Do you want to continue viewing report? [Y/N]: _

```

4.5.2 Control (While Loop)

```

while (1) {
    cout << "Enter your choice (Number 1 - 5 only): ";
    cin >> choiceLogin;

    if (choiceLogin == '1') {
        AdminLogin();
    }
    else if (choiceLogin == '2') {
        StaffLogin();
    }
    else if (choiceLogin == '3') {
        DonorLogin();
    }
    else if (choiceLogin == '4') {
        system("cls");
        ForgetPassword();
    }
    else if (choiceLogin == '5') {
        system("cls");
        main();
    }
    else {
        cout << "Invalid Choice! Only numeric number! Please enter again! ";
        cout << "\n";
        system("pause");
        MainLogin();
    }
}

```

```

LOGIN

[1] Login As Admin
[2] Login As Staff
[3] Login As Donor
[4] Forget Password
[5] Back to Main Menu

Enter your choice (Number 1 - 5 only): _

```

4.5.3 Control (Do-While Loop)

```

do
{
    cout << "-----" << endl;
    cout << " UPDATE RECORD MENU " << endl;
    cout << "-----" << endl;

    cout << "[D] Update Record of Donors" << endl;
    cout << "[H] Update Record of Hospital" << endl;
    cout << "[B] Update Record of Blood Stock" << endl;
    cout << "[S] Update Record of Donation" << endl;
    cout << "[M] Main Menu" << endl;

    cout << "\nPlease enter your choice (D, H, B, S, M): ";
    cin >> StaffUpdate;

    switch (StaffUpdate) {
    case 'H':
    case 'h':
        UpdateHospital();
        break;

    case 'D':
    case 'd':
        UpdateDonor();
        break;

    case 'B':
    case 'b':
        UpdateBlood();
        break;

    case 'S':
    case 's':

```

```

        UpdateDonation();
        break;

    case 'M':
    case 'm':
        StaffControlMain();
        break;

    default:
        cout << "Invalid choice!" << endl;
        system("pause");
        break;
    }
} while (StaffUpdate != 'D' || StaffUpdate != 'd' || StaffUpdate != 'b' ||
StaffUpdate != 'B' || StaffUpdate != 's' || StaffUpdate != 'S' || StaffUpdate !=
'h' || StaffUpdate != 'H' || StaffUpdate != 'M' || StaffUpdate != 'm');

```

```

-----
UPDATE RECORD MENU
-----
[D] Update Record of Donors
[H] Update Record of Hospital
[B] Update Record of Blood Stock
[S] Update Record of Donation
[M] Main Menu

Please enter your choice (D, H, B, S, M): _

```

4.5.4 Control (Jump statement with For loop and 'goto' clause)

```

for (int i = 0; i < *maxNum + 5; i++)
{
    cout << "-";
}
cout << ">" << endl;
cout << "\n" << endl;
goto confirmRpt1;

confirmRpt1:
    char continueRpt;
    cout << "Do you want to continue viewing report? [Y/N]: ";

```

```

cin >> continueRpt;

if (continueRpt == 'y' || continueRpt == 'Y')
{
    AdminReportMain();
}
else if (continueRpt == 'n' || continueRpt == 'N')
{
    AdminReportMain();
}
else
{
    cout << "Invalid choice!" << endl;
    goto confirmRpt1;
}

```

4.6 Array (Collecting data sets)

Count number of staff based on gender and store it into array. This is used to display the report in graphical method. The following is a code snippet for display number of staff based on position in graphical bar visual.

```

void AdminReportMain()
{
    system("cls");
    ShowTime();
    string numberStaff;
    int adminReport;
    cout << "-----" << endl;
    cout << " REPORT GENERATION FOR ADMIN " << endl;
    cout << "-----" << endl;

    string countAct_query = "Select COUNT(Staff_ID) FROM staff WHERE Active_Status
= 'Active'";
    const char* staf = countAct_query.c_str();
    qstate = mysql_query(conn, staf);
    if (!qstate)
    {
        res = mysql_store_result(conn);
        if (res->row_count == 1)
        {
            while (row = mysql_fetch_row(res))
            {
                numberStaff = row[0];
            }
        }
        else
        {
            cout << "Error" << endl;
            system("pause");
            AdminMainMenu(Staff_Name);
        }
    }
    else

```

```

{
    cout << "error!" << endl;
    system("pause");
    AdminMainMenu(Staff_Name);
}
cout << "\nThe default number of active staffs in the system: " << numberStaff
<< endl;

cout << "\nWhich attribute do you want to generate as report? " << endl;
cout << "[1] Gender" << endl;
cout << "[2] Position" << endl;
cout << "[3] Age" << endl;
cout << "Your choice: ";
cin >> adminReport;

else if (adminReport == 2) {
    string StaffPosition1, StaffPosition2, StaffPosition3;

    //calculate number of staff based on position
    //select a common single column but display it as different column
    //in this case, select from staff position column, but display as 3
columns
    //1. Number of staff that is doctor
    //2. Number of staff that is nurse
    //3. Number of staff that is staff
    //if there's any record then 1 (true)
    //else it's 0 (false) and return 0

    string countPos_query = "Select SUM(case when Staff_Position = 'Doctor'
then 1 else 0 end), SUM(case when Staff_Position = 'Nurse' then 1 else 0 end),
SUM(case when Staff_Position = 'Staff' then 1 else 0 end) FROM staff WHERE
Active_Status = 'Active'";
    const char* cu1 = countPos_query.c_str();
    qstate = mysql_query(conn, cu1);
    if (!qstate)
    {
        res = mysql_store_result(conn);
        if (res->row_count == 1)
        {
            while (row = mysql_fetch_row(res))
            {
                StaffPosition1 = row[0];
                StaffPosition2 = row[1];
                StaffPosition3 = row[2];
            }
        }
        else
        {
            cout << "Error" << endl;
        }
    }
    else
    {
        cout << "error!" << endl;
    }

    system("cls");
    ShowTime();
}

```

```

        cout << "\nThe number of active staff based on gender: " << endl;
        cout << "Doctor = " << foreground(Light_Red) << StaffPosition1 <<
foreground(White) << endl;
        cout << "Nurse = " << foreground(Light_Red) << StaffPosition2 <<
foreground(White) << endl;
        cout << "Staff = " << foreground(Light_Red) << StaffPosition3 <<
foreground(White) << endl << endl;

//convert the string data collected to integer data type
int pos1 = convertToInt(StaffPosition1);
int pos2 = convertToInt(StaffPosition2);
int pos3 = convertToInt(StaffPosition3);

        cout << "The following graph showing the number of staff based on gender:
" << endl;

//array for int calculated number of staff based on gender
int arrayPosition[3] = { pos1, pos2, pos3 };

//array used to print out the gender type
string arrayPostType[3] = { "Doctor", "Nurse", "Staff" };

//Display the graph
cout << "\n" << endl;
cout << background(Light_Red);
cout << "*****" << endl;
cout << "|          Graph (POSITION VS NUMBER OF STAFF)          |" << endl;
cout << "*****" << endl;
cout << background(Black);
cout << endl;
cout << setw(8) << "Position" << right << " Number of Staff" << endl;
cout << setw(10) << right << "/" << endl;

for (int i = 0; i < 3; i++)
{
    cout << setw(8) << left << arrayPostType[i] << left << "|";

    for (int j = 0; j < arrayPosition[i]; j++)
    {
        cout << "*";
    }
    if (arrayPosition[i] != 0) //showing the number of each gender in the
graph
    {
        cout << " " << foreground(Yellow) << arrayPosition[i] <<
foreground(White) << endl;
    }
    else
    {
        cout << endl;
    }
}

cout << setw(8) << right;
int* maxNum;
maxNum = max_element(arrayPosition, arrayPosition + 1);

```



```

    for (int i = 0; i < *maxNum + 5; i++)
    {
        cout << "-";
    }
    cout << ">" << endl;
    cout << "\n" << endl;
    goto confirmRpt2;

confirmRpt2:
    char continueRpt;
    cout << "Do you want to continue viewing report? [Y/N]: ";
    cin >> continueRpt;

    if (continueRpt == 'y' || continueRpt == 'Y')
    {
        AdminReportMain();
    }
    else if (continueRpt == 'n' || continueRpt == 'N')
    {
        AdminReportMain();
    }
    else
    {
        cout << "Invalid choice!" << endl;
        goto confirmRpt2;
    }
}

```

```

Date: 6/1/2023           Time: 3:00:25 pm

The number of active staff based on gender:
Doctor = 7
Nurse = 6
Staff = 7

The following graph showing the number of staff based on gender:

*****
|          Graph (POSITION VS NUMBER OF STAFF)          |
*****

Position Number of Staff
/ \
Doctor |***** 7
Nurse  |***** 6
Staff  |***** 7
----->

Do you want to continue viewing report? [Y/N]: _

```

4.7 Pointer

In this system, most pointers are used for making SQL statements. For example, if a user want to calculate a record, the system will use pointer to hold the SQL SELECT COUNT Statement to a constant character variable, then the character will make connection to SQL to execute the statement. The following sample code snippet is one of the count the number of staff where the age of staff is between 40 and 50:

```
string countAge3_query = "Select COUNT(Staff_ID) FROM staff WHERE Staff_Age >=40
AND Staff_Age < 50 AND Active_Status = 'Active'";
const char* cu3 = countAge3_query.c_str();
qstate = mysql_query(conn, cu3);
if (!qstate)
{
    res = mysql_store_result(conn);
    if (res->row_count == 1)
    {
        while (row = mysql_fetch_row(res))
        {
            StaffAge3 = row[0];
        }
    }
    else
    {
        cout << "Error" << endl;
        system("pause");
        AdminMainMenu(Staff_Name);
    }
}
else
{
    cout << "error!" << endl;
    system("pause");
    AdminMainMenu(Staff_Name);
}
```

4.8 Function

4.8.1 Function (Validation on Adding Records)

This function will receive the input from user and using INSERT operation for SQL to add the records into database. The following is the function that admin inserts records of staff to database:

```
void AdminAdd()
{
    system("cls");
    string Staff_ID, Staff_Name, Staff_Gender, Staff_Address, Staff_Telno,
    Staff_Position, Staff_Password, Admin_ID, Hospital_ID;
    int Staff_Age;
    char continueAdd;
    cout << background(Red) << foreground(Black);
```

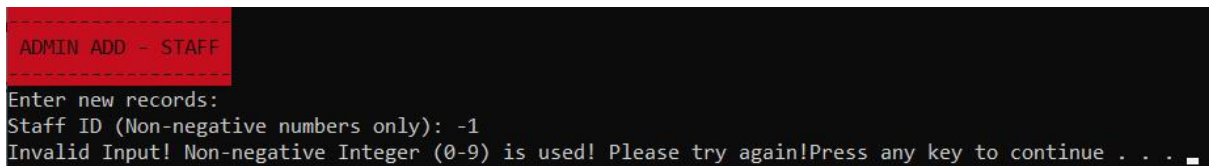
```

cout << "-----" << endl;
cout << " ADMIN ADD - STAFF " << endl;
cout << "-----" << endl;
cout << background(Black) << foreground(White) << resetANSI();
cout << "Enter new records: " << endl;
cout << "Staff ID (Non-negative numbers only): ";

int IDNum;
cin >> IDNum;

string StID;
if (IDNum >= 0 && IDNum < 10)
{
    StID.append("ST00");
    StID.append(to_string(IDNum));
}
else if (IDNum >= 10 && IDNum < 100)
{
    StID.append("ST0");
    StID.append(to_string(IDNum));
}
else if (IDNum >= 100 && IDNum < 1000)
{
    StID.append("ST");
    StID.append(to_string(IDNum));
}
else if (IDNum >= 1000 && IDNum < 10000)
{
    StID.append("ST");
    StID.append(to_string(IDNum));
}
else
{
    cout << "Invalid Input! Non-negative Integer (0-9) is used! Please try
again!";
    system("pause");
    system("cls");
    AdminControlMain();
}

```



```

ADMIN ADD - STAFF
Enter new records:
Staff ID (Non-negative numbers only): -1
Invalid Input! Non-negative Integer (0-9) is used! Please try again!Press any key to continue . . .

```

4.8.2 Function (Display own account information)

This function will display the user's own account information after they have made successful login. The following is one of the examples for displaying staff account:

```
void StaffAccount(string id)
```

```

{
    system("cls");
    ShowTime();
    cout << "===== " << endl;
    cout << "          ACCOUNT INFORMATION          " << endl;
    cout << "===== " << endl;

    string search_query = "SELECT Staff_ID, Staff_Name, Staff_Gender, Staff_Age,
Staff_Address, Staff_Telno, Staff_Position FROM staff WHERE Staff_ID = '" + id +
"' AND Active_Status = 'Active'";
    const char* q = search_query.c_str();
    qstate = mysql_query(conn, q);
    if (!qstate)
    {
        res = mysql_store_result(conn);
        while (row = mysql_fetch_row(res))
        {
            cout << "Staff ID: " << row[0] << endl;
            cout << "Name: " << row[1] << endl;
            cout << "Gender: " << row[2] << endl;
            cout << "Age: " << row[3] << endl;
            cout << "Address: " << row[4] << endl;
            cout << "Phone: " << row[5] << endl;
            cout << "Position: " << row[6] << endl;
        }
        system("pause");
        system("cls");
        StaffMainMenu(id, Staff_Name);
    }
    else
    {
        cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
        system("pause");
        system("cls");
        StaffMainMenu(id, Staff_Name);
    }
}

```

```

Date: 6/1/2023          Time: 3:05:17 pm

=====
ACCOUNT INFORMATION
=====
Staff ID: ST0001
Name: Lee Guan Eng
Gender: M
Age: 25
Address: 13, Jalan Melanor, Taman Mentor, 73500 Petiling
Phone: 017-4702345
Position: Doctor
Press any key to continue . . .

```

4.8.3 Function (Deleting records)

In real-time environments, only privileged panels have the ability to delete the records in database. In this case, only soft-delete is done by updating the status to inactive so that the records seems to be deleted from user's view but actually the 'deleted' records are still remained in the database. The sql command to delete the record will be put as comment. The following is an example of deleting a blood donation record:

```
void DeleteDonation()
{
    system("cls");
    string Donation_ID;
    char confirmDel, continueDel;
    cout << foreground(Cyan);
    cout << "-----" << endl;
    cout << " DELETE RECORD - DONATION " << endl;
    cout << "-----" << endl;
    cout << background(Black) << foreground(White) << resetANSI();
    cout << "\nEnter Donation ID to search: ";
    cin >> Donation_ID;

    cout << "\nHere's the record found: \n" << endl;
    string search_query = "SELECT t.Donation_ID, t.Donation_Date,
t.Donation_Volume, r.Record_ID, h.Hospital_ID, d.Donor_ID FROM blood_donation t,
donor_record r, hospital h, donor d WHERE t.Donation_ID = '" + Donation_ID + "'
AND t.Record_ID = r.Record_ID AND r.Hospital_ID = h.Hospital_ID AND r.Donor_ID =
d.Donor_ID ";
    const char* q = search_query.c_str();
    qstate = mysql_query(conn, q);
    if (!qstate)
    {
        res = mysql_store_result(conn);
        while (row = mysql_fetch_row(res))
        {
            cout << "Donation ID: " << row[0] << endl;
            cout << "Date: " << row[1] << endl;
            cout << "Volume: " << row[2] << endl;
            cout << "Record ID: " << row[3] << endl;
            cout << "Hospital ID: " << row[4] << endl;
            cout << "Donor ID: " << row[5] << endl;
        }
    }
    else
    {
        cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
        system("pause");
        DeleteDonation();
    }

    cout << "Are you confirm to delete this record [Y/N]: ";
    cin >> confirmDel;

    if (confirmDel == 'Y' || confirmDel == 'y')
    {
```

```

        //string delete_query = "DELETE FROM blood_donation where Donation_ID = '"
+ Donation_ID + "'";
        //const char* q = delete_query.c_str();

        string update_query = "UPDATE blood_donation SET Status = 'Unavailable'
WHERE Donation_ID = '" + Donation_ID + "' ";
        const char* q = update_query.c_str();
        qstate = mysql_query(conn, q);
        cout << "Successfully delete a blood donation record! " << endl;
        cout << "Do you want to continue deleting record?[Y/N]: ";
        cin >> continueDel;
        if (continueDel == 'y' || continueDel == 'Y')
            DeleteDonation();
        else if (continueDel == 'n' || continueDel == 'N')
            StaffControlDelete();
    }
    else
    {
        cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
        system("pause");
        StaffControlDelete();
    }
}

```

```

-----
DELETE RECORD - DONATION
-----

Enter Donation ID to search: DT0013

Here's the record found:

Donation ID: DT0013
Date: 2020-04-30
Volume: 450
Record ID: RC0013
Hospital ID: HS0007
Donor ID: DN0002
Are you confirm to delete this record [Y/N]: Y
Successfully delete a blood donation record!
Do you want to continue deleting record?[Y/N]: _

```

4.9 Calculation

This function will enable auto calculation for donor's age once the donor input their own date of birth and compare the date with current date on real-time system. It will return the age back to the function call.

```

int calculateAge(int year, int month, int day)
{
    //getting current date

```

```

int currYear, currMonth, currDay;
int age = 0;

time_t now = time(0);
tm* ltm = localtime(&now);

currYear = 1900 + ltm->tm_year;
currMonth = 1 + ltm->tm_mon;
currDay = ltm->tm_mday;

//getting the age by comparing current date and user input date
if (currMonth < month)
    age = currYear - year;

    if (currDay < day)
        age = currYear - year - 1;
    else
        age = currYear - year;

return age;
}

```

4.10 Error Handling

Every process that involves the use of a database will be handled to ensure that the queries are successfully performed. Below is an example of the implementation for searching a donor record based on Donor ID:

```

system("cls");
cout << foreground(Cyan);
cout << "-----" << endl;
cout << " SEARCH RECORD - DONOR " << endl;
cout << "-----" << endl;
cout << "\n";
cout << background(Black) << foreground(White) << resetANSI();
cout << "Please select the attribute you want to search: " << endl;
cout << "[1] Donor ID " << endl;
cout << "[2] Name " << endl;
cout << "[3] Gender " << endl;
cout << "[4] Age " << endl;
cout << "[5] Date of Birth " << endl;
cout << "[6] Address " << endl;
cout << "[7] Blood Type " << endl;
cout << "[8] Height " << endl;
cout << "[9] Weight " << endl;
cout << "[10] Phone " << endl;
cout << "[11] Back to Main Menu " << endl;

cout << "\nYour Choice >> ";
cin >> searchDonor;

if (searchDonor == 1)
{

```

```

cout << "Enter Donor ID to search (Format: DN****): ";
cin >> Donor_ID;

cout << "\nHere's the record found: \n" << endl;
string search_query = "SELECT Donor_ID, Donor_Name, Donor_Gender,
Donor_Age, Donor_DOB, Donor_Address, Donor_BloodType, Donor_Height, Donor_Weight,
Donor_TelNo FROM donor WHERE Donor_ID = '" + Donor_ID + "' AND Active_Status =
'Active'";
const char* q = search_query.c_str();
qstate = mysql_query(conn, q);
if (!qstate)
{
    res = mysql_store_result(conn);
    while (row = mysql_fetch_row(res))
    {
        cout << "Donor ID: " << row[0] << endl;
        cout << "Name: " << row[1] << endl;
        cout << "Gender: " << row[2] << endl;
        cout << "Age: " << row[3] << endl;
        cout << "Date of Birth: " << row[4] << endl;
        cout << "Address: " << row[5] << endl;
        cout << "Blood Type: " << row[6] << endl;
        cout << "Height: " << row[7] << endl;
        cout << "Weight: " << row[8] << endl;
        cout << "Phone: " << row[9] << endl;
        cout << endl;
    }
    cout << endl << "Do you want to search other donor?[Y/N]: ";
    cin >> SearchDon;
    if (SearchDon == 'y' || SearchDon == 'Y')
        SearchDonor();
    else if (SearchDon == 'n' || SearchDon == 'N')
        StaffControlMain();
}
else
{
    cout << "There's no any result about staff found! " << endl;
    cout << "Query Execution Problem!" << mysql_errno(conn) << endl;
    system("pause");
    StaffControlSearch();
}
}

```

The code segment above shows the error handling of the SQL query that will come out of the output with null results for donor records. Thus, selection if-else is applied to control this situation happens when the result set of the query is null or empty, it will prompt out the message by printing the output message to the users.


```
-----  
SEARCH RECORD - DONOR  
-----  
  
Please select the attribute you want to search:  
[1] Donor ID  
[2] Name  
[3] Gender  
[4] Age  
[5] Date of Birth  
[6] Address  
[7] Blood Type  
[8] Height  
[9] Weight  
[10] Phone  
[11] Back to Main Menu  
  
Your Choice >> 1  
Enter Donor ID to search (Format: DN****): 18  
  
Here's the record found:  
  
Do you want to search other donor?[Y/N]:
```

CHAPTER 5 CONCLUSION

5.1 Introduction

Blood Donation Management System provides sufficient support for administrators to manage the details of staffs in efficient way, while staff can manage the information of hospital, donors, blood stock and blood donation records in a better way. This system also helps donors to have a look at their own donation records. This is because authentication such as login is included in the system so that only privileged person is able to manipulate the data.

By developing this system, it enables staff to make queries for selecting certain period to view all the donation records during the selected period. It will also calculate the total records during that period. Besides, this system helps to display a report to calculate the number of staff from the admin's perspective so that they can make good management on staffs. It provides an account system for staff and donors to view their own personal information. Besides, forget password module enable each user to retrieve back their account so that they wont have to register again once their ID exist based on the database.

However, there are some problems and limitations that exist while developing the system that the data integrity in the system must be always controlled at high level so that the authorization for user is working well and interact with the interrelated data. Other than that, there is some information that are private to other user. For example, only admin can view the information of staffs records and donors can only view their own donation records as the manipulation of donors' information is done at staff level. A lot of confidential records are required to integrate each user (refers to staff and donor) so that the records are accessible by only personnel and hence the login system needs to be keep secured from time to time.

5.2 Suggestion

As mentioned above, there are some suggestions to enhance the system in a better way by adding a notification ability so that this system can notify respective users once there's any update of information related to them. Next, another important point needed to be enhanced is that some private personal data such as contact number and personal must be encrypted so that protect the privacy of the user also increase the system's security and robustness to hackers or 47 attackers.

The system can also be enhanced by inserting a security program that allows only 3 times of login per time to avoid brute force attacks. This prevents any user wants to try login other user accounts so that they want to hack other user account.

Other than that, an accounting function also can be included in this system so that each user can keep track of histories of actions such as which staff have recently logged in and their own actions on manipulate the data. This helps administrators to keep track of staff's actions and prevent any private information exposure.

REFERENCES

1. “Recursive Relationships in ER Diagrams - GeeksforGeeks.” *GeeksforGeeks*, 18 Jan. 2018, www.geeksforgeeks.org/recursive-relationships-in-er-diagrams.
2. *Database entities in E/R Modeling*. Database Management. (n.d.). Retrieved January 6, 2023, from https://databasemanagement.fandom.com/wiki/Database_Entities_in_E/R_Modeling
3. “SQL Joins.” *SQL Joins*, www.w3schools.com/sql/sql_join.asp. Accessed 6 Jan. 2023.
4. “SQL Equijoin - W3resource.” *W3resource*, www.w3resource.com/sql/joins/perform-an-equi-join.php. Accessed 6 Jan. 2023.
5. “SQL | EQUI Join and NON EQUI JOIN - GeeksforGeeks.” *GeeksforGeeks*, 5 Sept. 2020, www.geeksforgeeks.org/sql-equi-join-and-non-equi-join.
6. “C++ Arrays (With Examples).” *C++ Arrays (With Examples)*, www.programiz.com/cpp-programming/arrays. Accessed 6 Jan. 2023.