EX NO:1	**WRITE THE COMPLETE PROBLEM STATEMENT**
DATE:	

AIM:

To prepare a PROBLEM STATEMENT for a blood bank management system.

ALGORITHM:

- The problem statement is the initial starting point for a project.
- A problem statement describes what needs to be done without describing how.
- It is generally a one-to-three-page document that all project stakeholders agree upon, describing the goals of the project at a high level.
- The problem statement is intended for a broad audience and should be written in non-technical terms.
- It helps both technical and non-technical personnel communicate effectively by providing a clear description of the problem.
- The problem statement does not describe the solution to the problem.

INPUT:

- The input to requirement engineering is the problem statement prepared by the customer.
- It may include an overview of the existing system and the broad expectations from the new system.
- The first phase of requirements engineering begins with requirements elicitation, i.e., gathering information about the requirements.

  Here, requirements are identified with the help of the customer and existing system processes.

Problem:

Current blood bank operations are plagued by inefficiencies in manual record-keeping, lack of real-time blood inventory updates, and difficulties in locating suitable donors during emergencies. These issues lead to delays in critical medical situations and increase the risk of medical complications. A robust Blood Bank Management System can automate processes, provide accurate real-time data, and enable faster donor-recipient matching, ultimately improving the efficiency and effectiveness of blood bank services.

Background:

Blood is an essential component in modern medical treatments, used in various procedures like surgeries, trauma care, cancer treatments, and for patients with blood disorders. Blood banks play a crucial role in ensuring that blood is readily available for patients in need. However, blood banks face significant challenges, such as managing donor information, ensuring blood stock levels are optimal, and efficiently matching donors with recipients in emergencies.

Traditionally, blood banks rely on manual record-keeping systems and outdated inventory management methods, which can lead to errors, delays, and inefficiencies. With the increasing demand for blood, especially during crises, it becomes essential to modernize the way blood banks operate.

A **Blood Bank Management System (BBMS)** aims to digitize and streamline operations, allowing for more efficient management of donors, blood inventory, and patient requirements. By integrating technology into these processes, blood banks can become more responsive, accurate, and capable of saving lives.

Relevance:

The relevance of a Blood Bank Management System is rooted in its ability to address the critical challenges faced by healthcare institutions. Blood banks need to maintain accurate records of donors, blood inventory, and patient requirements while ensuring that blood is available in times of need. Inadequate management can result in:

- **Inventory Shortages:** Inaccurate blood stock tracking may lead to insufficient supply during emergencies.
- **Donor Mismatch:** A lack of a proper matching system can result in delays and mistakes, potentially causing harm to patients.
- **Operational Inefficiencies:** Manual systems are prone to errors, and time-consuming paperwork may reduce the efficiency of the blood donation process.
- **Lack of Real-Time Data:** Without immediate access to updated information, healthcare providers may not know the current blood supply levels, leading to possible delays in urgent medical cases.

In this context, a comprehensive **BBMS** is highly relevant. It ensures that blood banks can respond to medical demands promptly, track inventory efficiently, maintain accurate donor and recipient records, and improve overall operational effectiveness.

Objectives:

The main objectives of a **Blood Bank Management System** are to:

1. **Automate Donor Information Management**:
   - Digitally store donor profiles, including personal details, blood type, donation history, and contact information.
   - Provide a simple interface for blood donors to update their personal information and track donation history.
2. **Real-Time Blood Inventory Management**:
   - Track blood stock levels in real-time, including the number of units of each blood group available.
   - Notify administrators when stock levels are low or when blood nearing its expiration date needs to be used or discarded.
3. **Efficient Donor-Recipient Matching**:
   - Develop an automated system that quickly matches available blood donors with patients who require specific blood types.
   - Provide a search feature to find potential donors based on blood type and availability..
4. **Provide Data Analytics and Reporting**:
   - Offer detailed reports on donor activity, blood inventory levels, and usage trends.
   - Provide insights into donor demographics and donation patterns to help plan future drives and optimize blood collection.
5. **Ensure Compliance with Medical Standards**:
   - Ensure that the system adheres to legal and medical standards for blood donation, storage, and distribution.
   - Maintain accurate records that comply with healthcare regulations and are ready for audits.
6. **Enhance Emergency Preparedness**:
   - Provide quick access to information on available blood supplies, especially in crisis situations (e.g., accidents, natural disasters).
   - Allow for immediate decision-making regarding blood usage and donor mobilization.

Result:

EX NO:2	**WRITE THE SOFTWARE REQUIREMENT SPECIFICATION DOCUMENT**
DATE	

AIM:

To do requirement analysis and develop Software Requirement Specification Sheet (SRS) for blood bank management system.

ALGORITHM:

SRS shall address are the following:

a) **Functionality.** What is the software supposed to do?

b) **External interfaces.** How does the software interact with people, the system‘s hardware, other hardware, and other software?

c) **Performance.** What is the speed, availability, response time, recovery time of various software functions, etc.?

d) **Attributes.** What is the portability, correctness, maintainability, security, etc. considerations?

e) **Design constraints imposed on an implementation.** Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.

1. Introduction

- **1.1 Purpose:**
  Define the purpose of the SRS document, outlining its intended audience and the goals of the blood bank management system.
- **1.2 Scope:**
  Describe the scope of the blood bank management system, including the major functionalities such as Donor Registration, Inventory Management, Blood Donation Management, Security.
- **1.3 Definitions, Acronyms, and Abbreviations:**
  List all key terms, abbreviations, and acronyms used throughout the document.
- **1.4 References:**
  Include any external documents, standards, or regulations that the system must comply with (e.g., legal standards for blood bank management system).

---

2. Overall Description

- **2.1 Product Perspective:**
  Describe the overall system architecture, including how the blood bank management system fits into the existing electoral framework (if applicable).

- **2.2 Product Features:**
  Provide an overview of the core features of the system, such as Donor Registration, Inventory Management, Blood Donation Management.
- **2.3 User Classes and Characteristics:**
  Identify the different user types (e.g., Donors, Healthcare Providers/Medical Staff, Blood Bank Administrators) and their needs and permissions.
- **2.4 Operating Environment:**
  Describe the technical environment (e.g., web platform, mobile app, server specifications, operating systems) in which the blood bank management system will operate.
- **2.5 Design and Implementation Constraints:**
  Identify any constraints on the system design, such as compliance with regulations, security standards, or technological limitations.
- **2.6 Assumptions and Dependencies:**
  List assumptions made during system development and any external dependencies (e.g., third-party APIs for weather data, election scheduling).

---

3. System Features

- **3.1 Feature 1: Donor Registration and Management**
  - **Description:** Detailed description of how users will register and authenticate their identity.
  - **Functional Requirements:** Registration process, authentication methods (password, two-factor authentication), and eligibility verification.
- **3.2 Feature 2: Blood Donation Scheduling**
  - **Description:** Donors can schedule a donation appointment, and the system tracks donation history.
  - **Functional Requirements:** Appointment booking, encryption, ability for donors to cancel or reschedule.
- **3.3 Feature 3: Blood Inventory Management**
  - **Description:** Tracks available blood units, expiration dates, and conditions for storage.
  - **Functional Requirements:** Real-time blood stock tracking, Alerts when blood stock levels are low and ability to update blood stock after donations or usage.
- **3.4 Feature 4: Security and Privacy**
  - **Description:** Measures to protect donor data and prevent unauthorized access.
  - **Functional Requirements:** Data encryption, authentication protocols, secure communication channels, and data anonymization.

---

4. External Interface Requirements

- **4.1 User Interfaces:**
  The system will have a simple, user-friendly interface for donors, medical staff, and administrators.

- **4.2 Hardware Interfaces:**
  Specify any hardware that the system will interact with (e.g., biometric scanners for authentication, barcode scanners, etc.).
- **4.3 Software Interfaces:**
  Hospital Management Systems: Integrate with hospital software for patient blood requests.
- **4.4 Communication Interfaces:**
  Define how the system will communicate over networks (e.g., HTTP, SSL/TLS encryption for secure communication).

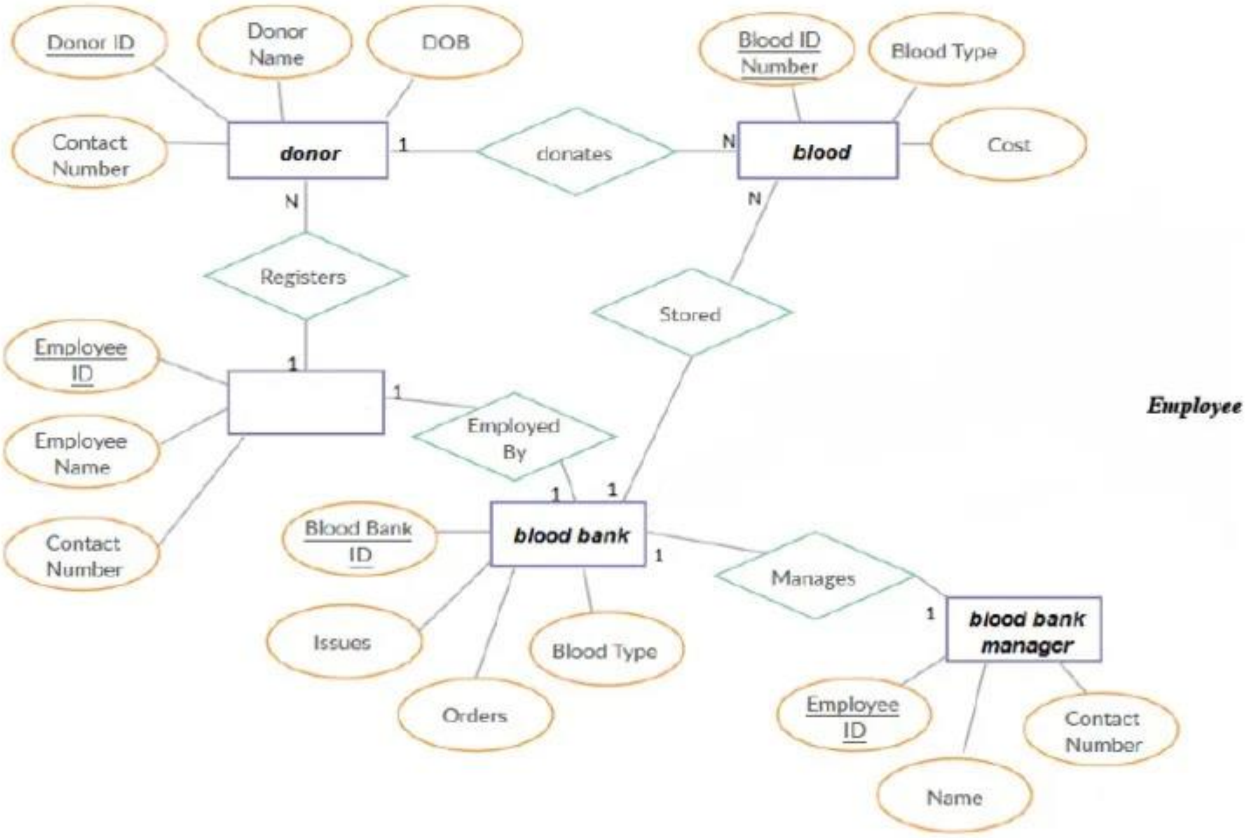
---

5. System Attributes

- **5.1 Performance Requirements:**
  The system should be capable of handling at least 1,000 concurrent users.
- **5.2 Security Requirements:**
  Specify the security standards and features required for the system, including encryption, audit trails, and protection against cyberattacks (e.g., DDoS).
- **5.3 Reliability:**
  Define expected system uptime, fault tolerance, and backup/recovery requirements.
- **5.4 Availability:**
  Specify the availability requirements, particularly during election periods (e.g., 24/7 availability with minimal downtime).

Result:

Blood Donation Management System ER Diagram



EX NO:3	**DRAW THE ENTITY RELATIONSHIP DIAGRAM**
DATE	

AIM:

To Draw the Entity Relationship Diagram for blood bank management system.

ALGORITHM:

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

Step 4: Mapping of Binary 1:N Relationship Types.

Step 5: Mapping of Binary M:N Relationship Types.

Step 6: Mapping of Multivalued attributes.

INPUT:

Entities

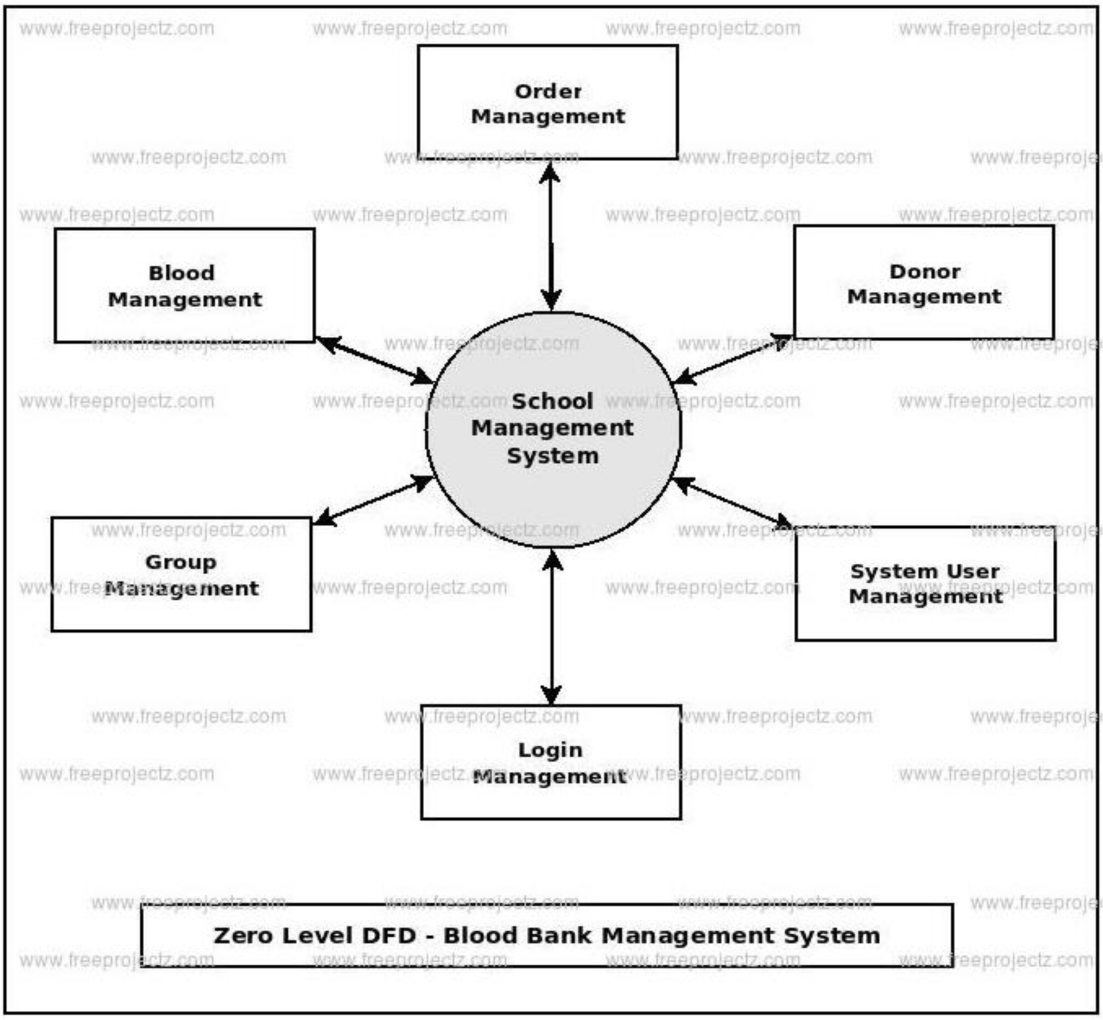
Entity Relationship Matrix

Primary Keys

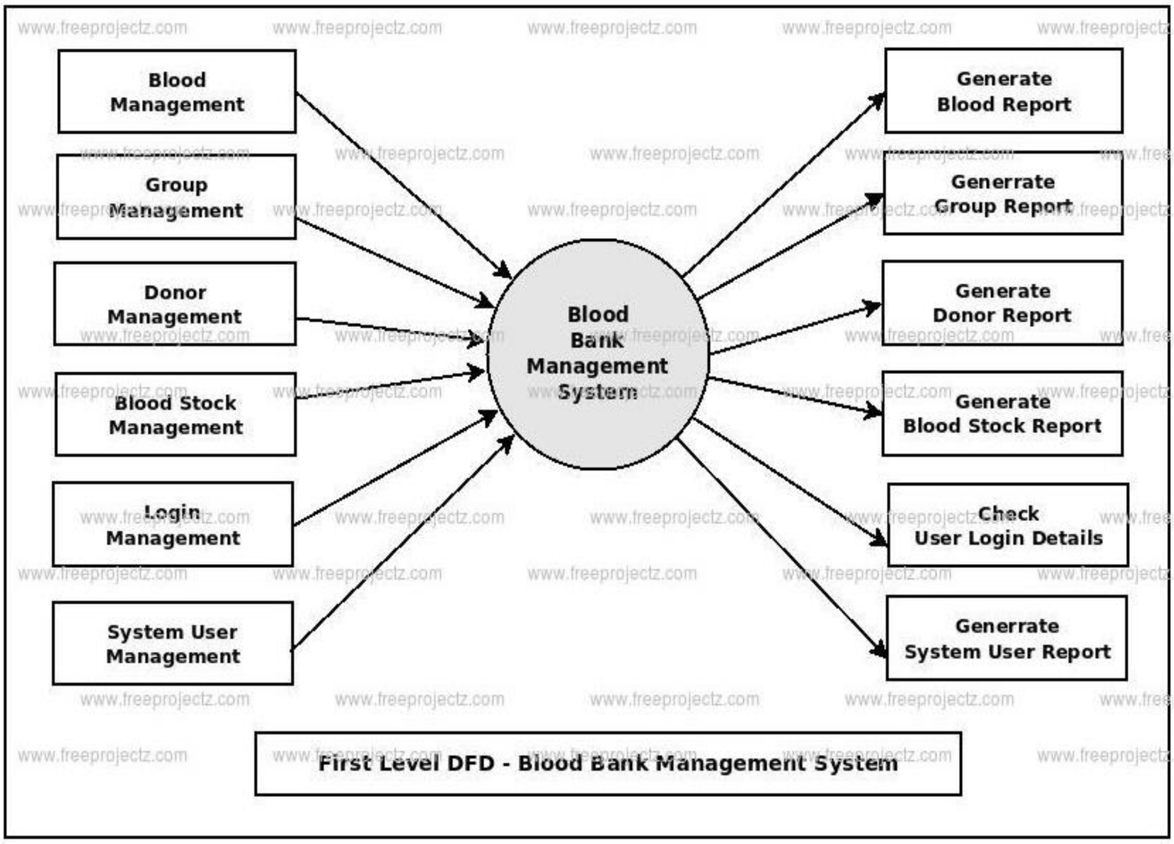
Attributes

Mapping of Attributes with Entities

Result:



Zero Level DFD - Blood Bank Management System



First Level DFD - Blood Bank Management System

EX NO:4	**DRAW THE DATA FLOW DIAGRAMS AT LEVEL 0 AND LEVEL 1**
DATE	

AIM:

To Draw the Data Flow Diagram for blood bank management system and list the Modules in the Application.

ALGORITHM:

1. Open the Visual Paradigm to draw DFD (Ex.Lucidchart)
2. Select a data flow diagram template
3. Name the data flow diagram
4. Add an external entity that starts the process
5. Add a Process to the DFD
6. Add a data store to the diagram
7. Continue to add items to the DFD
8. Add data flow to the DFD
9. Name the data flow
10. Customize the DFD with colours and fonts
11. Add a title and share your data flow diagram

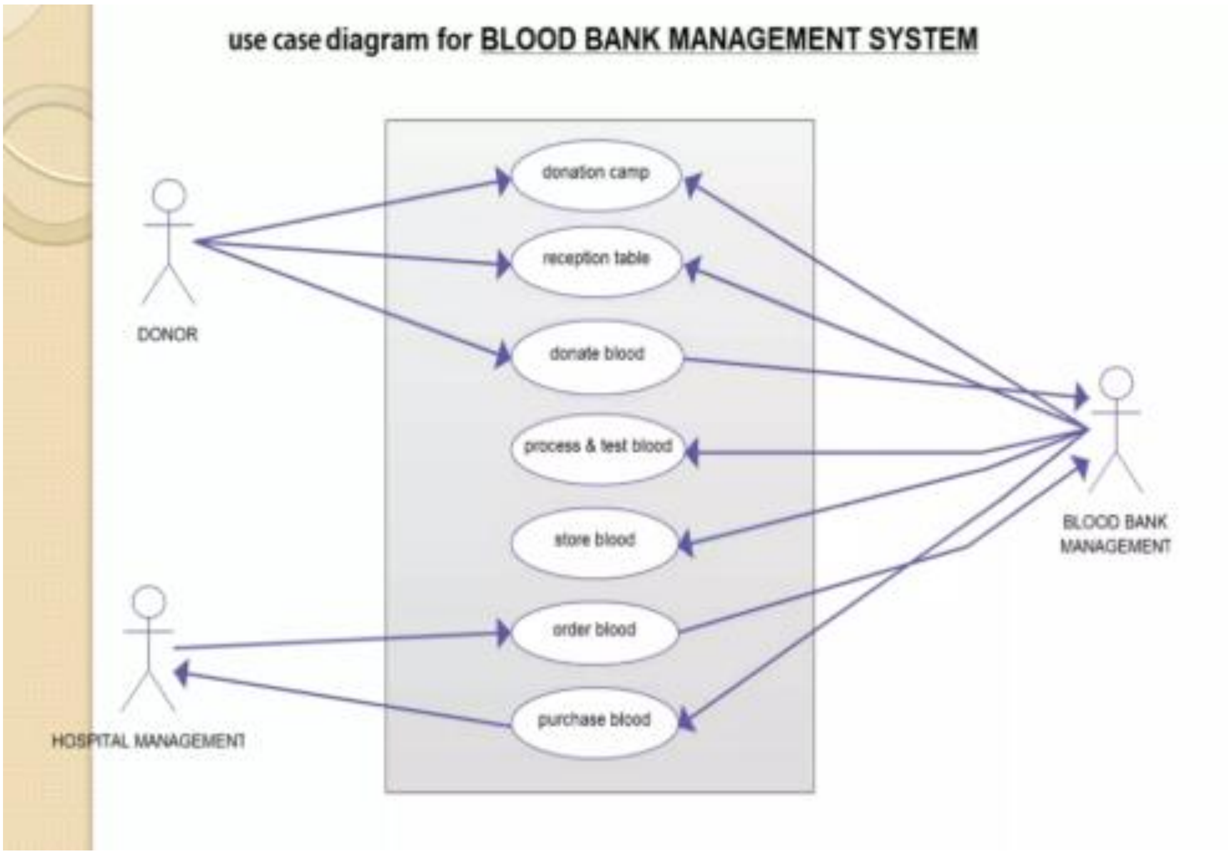
INPUT:

Processes

Datastores

External Entities

Result:



use case diagram for **BLOOD BANK MANAGEMENT SYSTEM**

EX NO:5	**DRAW USE CASE DIAGRAM**
DATE	

AIM:

To Draw the Use Case Diagram for blood bank management system.

ALGORITHM:

Step 1: Identify Actors

Step 2: Identify Use Cases

Step 3: Connect Actors and Use Cases

Step 4: Add System Boundary

Step 5: Define Relationships

Step 6: Review and Refine

Step 7: Validate

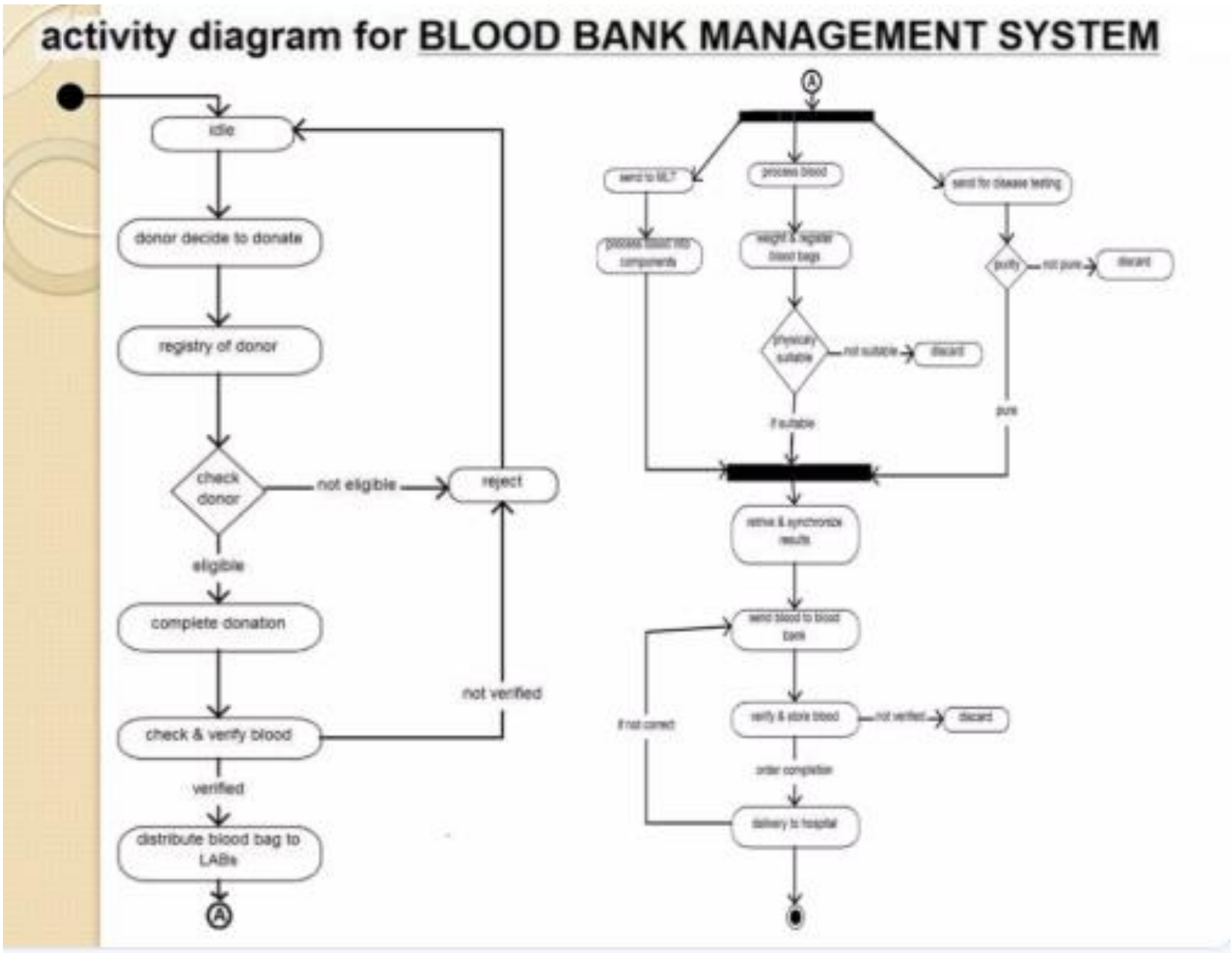
INPUTS:

Actors

Use Cases

Relations

Result:



activity diagram for BLOOD BANK MANAGEMENT SYSTEM

EX NO:6	**DRAW ACTIVITY DIAGRAM OF ALL USE CASES.**
DATE	

AIM:

To draw the activity Diagram for blood bank management system.

ALGORITHM:

Step 1: Identify the Initial State and Final States

Step 2: Identify the Intermediate Activities Needed

Step 3: Identify the Conditions or Constraints

Step 4: Draw the Diagram with Appropriate Notations

INPUTS:

Activities

Decision Points

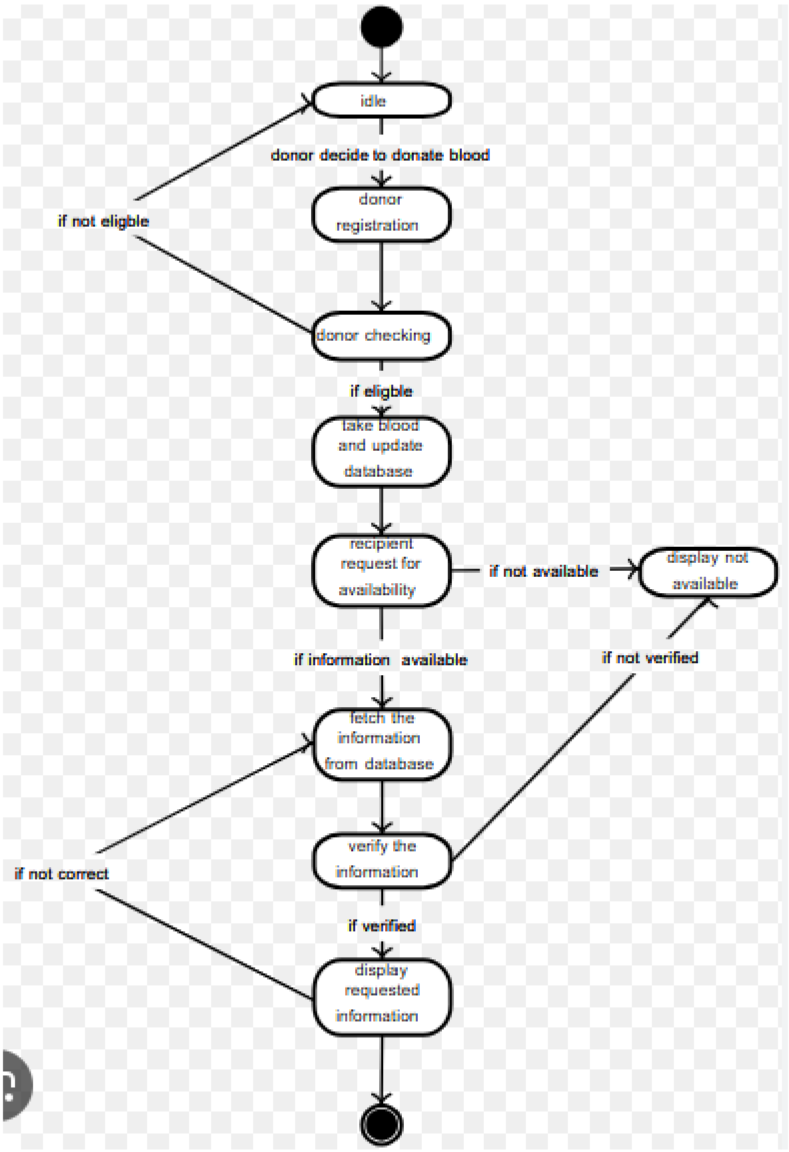
Guards

Parallel Activities

Conditions

Result:

STATE CHART DIAGRAM



EX NO:7	**DRAW STATE CHART DIAGRAM OF ALL USE CASES.**
DATE	

AIM:

To Draw the State Chart Diagram for blood bank management system.

ALGORITHM:

STEP-1: Identify the important objects to be analysed.

STEP-2: Identify the states.

STEP-3: Identify the events.

INPUTS:

Objects

States

Events

Result:

EX NO:8	DRAW SEQUENCE DIAGRAM OF ALL USE CASES.
DATE	

AIM:

To Draw the Sequence Diagram for blood bank management system.

ALGORITHM:

1. Identify the Scenario
2. List the Participants
3. Define Lifelines
4. Arrange Lifelines
5. Add Activation Bars
6. Draw Messages
7. Include Return Messages
8. Indicate Timing and Order
9. Include Conditions and Loops
10. Consider Parallel Execution
11. Review and Refine
12. Add Annotations and Comments
13. Document Assumptions and Constraints
14. Use a Tool to create a neat sequence diagram

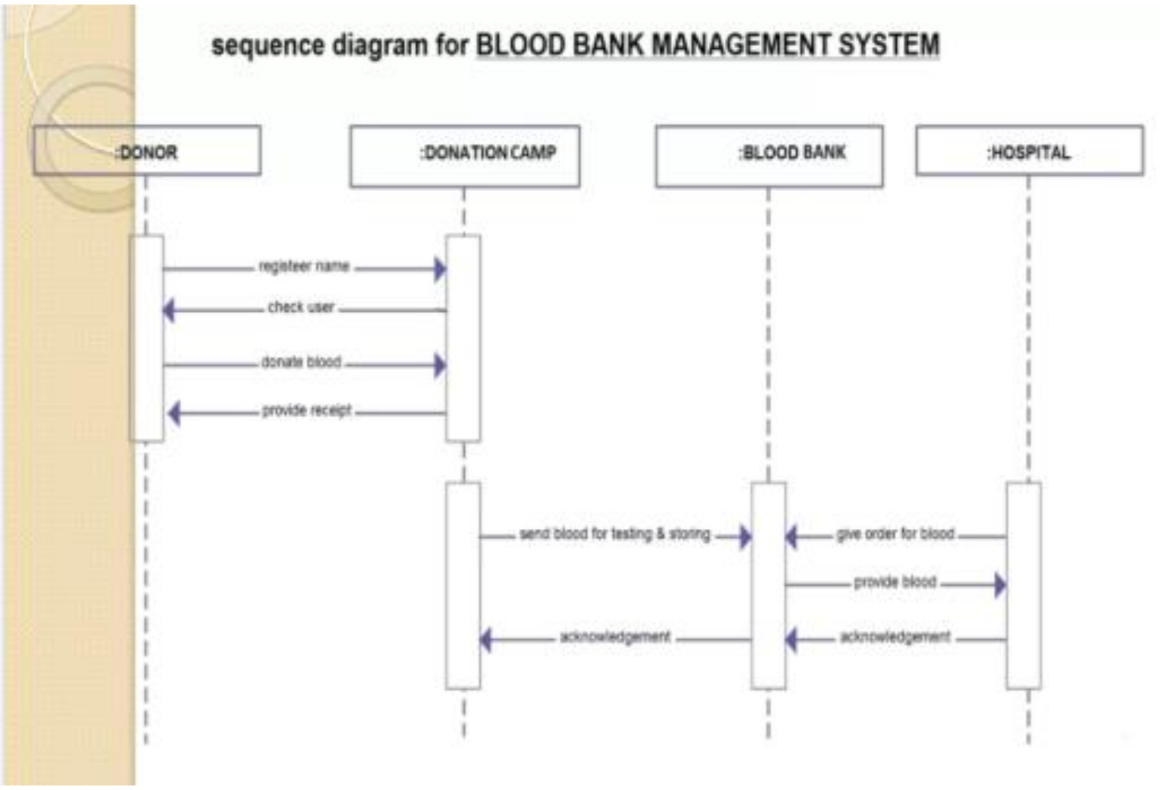
INPUTS:

Objects taking part in the interaction.

Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

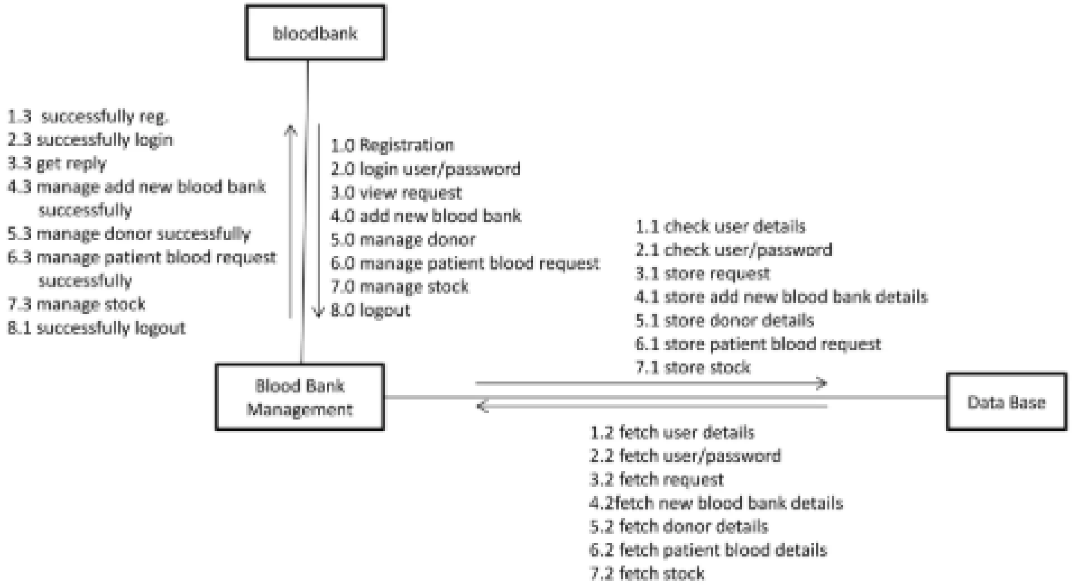


sequence diagram for BLOOD BANK MANAGEMENT SYSTEM

Result:

COLLABORATION DIAGRAM

Blood bank:



EX NO:9	**DRAW COLLABORATION DIAGRAM OF ALL USE CASES**
DATE	

AIM:

To Draw the Collaboration Diagram for blood bank management system.

ALGORITHM:

Step 1: Identify Objects/Participants

Step 2: Define Interactions

Step 3: Add Messages

Step 4: Consider Relationships

Step 5: Document the collaboration diagram along with any relevant

explanations or annotations.

INPUTS:

Objects taking part in the interaction.

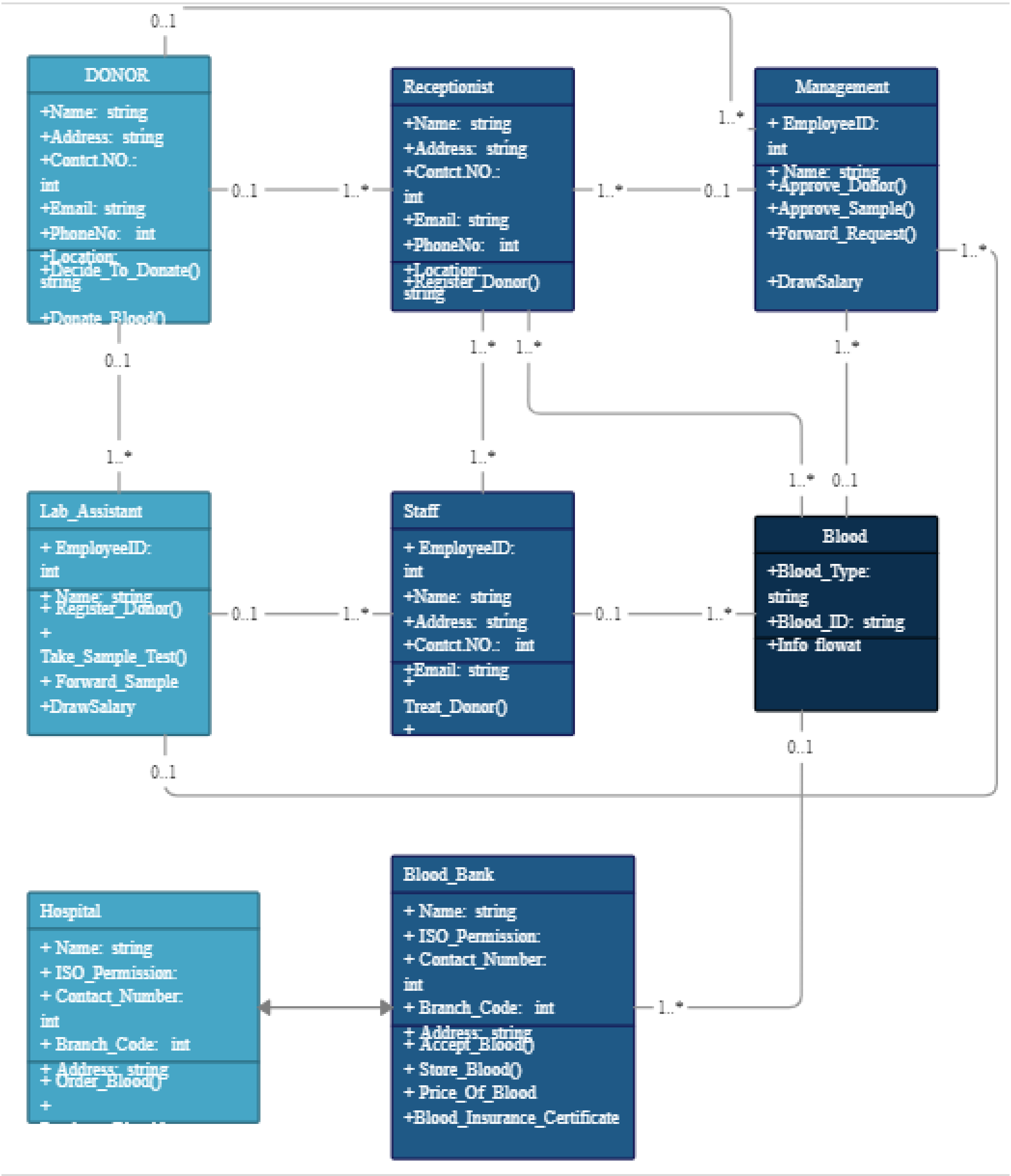
Message flows among the objects.

The sequence in which the messages are flowing.

Object organization.

Result:

CLASS DIAGRAM



EX NO:10	**ASSIGN OBJECTS IN SEQUENCE DIAGRAM TO CLASSES AND MAKE CLASS DIAGRAM.**
DATE	

AIM:

To Draw the Class Diagram for blood bank management system.

ALGORITHM:

1. Identify Classes
2. List Attributes and Methods
3. Identify Relationships
4. Create Class Boxes
5. Add Attributes and Methods
6. Draw Relationships
7. Label Relationships
8. Review and Refine
9. Use Tools for Digital Drawing

INPUTS:

1. Class Name
2. Attributes
3. Methods
4. Visibility Notation

RESULT:

EX NO:11	**MINI PROJECT- BLOOD BANK MANAGEMENT SYSTEM**
DATE	

AIM:

The primary aim of this mini-project is to develop a secure and user-friendly blood bank management system. By utilizing Streamlit for a seamless user interface, we aim to enhance the donor process, ensuring transparency, efficiency, and donor confidentiality.

ALGORITHM:

1. User registers with valid credentials.

2. User logs in using their credentials.

3. System displays a list of donors.

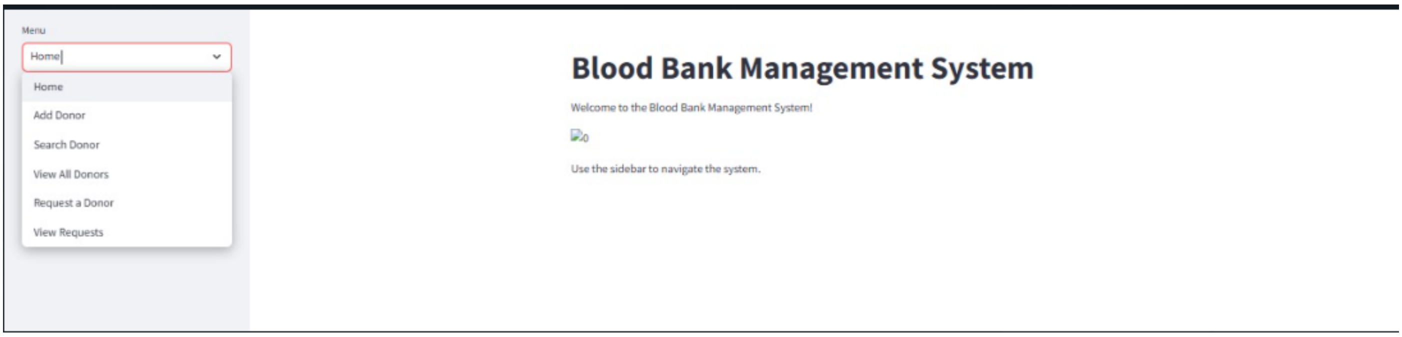
4. User selects their blood group and donate.

5. User's donation is encrypted and stored securely.

6. Donations are noted and stored successfully.

PROGRAM:

```
IMPORT STREAMLIT AS ST

IMPORT PANDAS AS PD

IF 'DONORS' NOT IN ST.SESSION_STATE:

    ST.SESSION_STATE['DONORS'] = []

IF 'REQUESTS' NOT IN ST.SESSION_STATE:

    ST.SESSION_STATE['REQUESTS'] = []

ST.TITLE("BLOOD BANK MANAGEMENT SYSTEM")

MENU = ST.SIDEBAR.SELECTBOX("MENU", ["HOME", "ADD DONOR", "SEARCH DONOR", "VIEW ALL DONORS", "REQUEST A DONOR", "VIEW REQUESTS"])

IF MENU == "HOME":

    ST.WRITE("WELCOME TO THE BLOOD BANK MANAGEMENT SYSTEM!")

    ST.IMAGE("HTTPS://IMAGES.UNSPLASH.COM/PHOTO-1597764692371-9B4BCE3D917C", WIDTH=700)

    ST.WRITE("USE THE SIDEBAR TO NAVIGATE THE SYSTEM.")
```

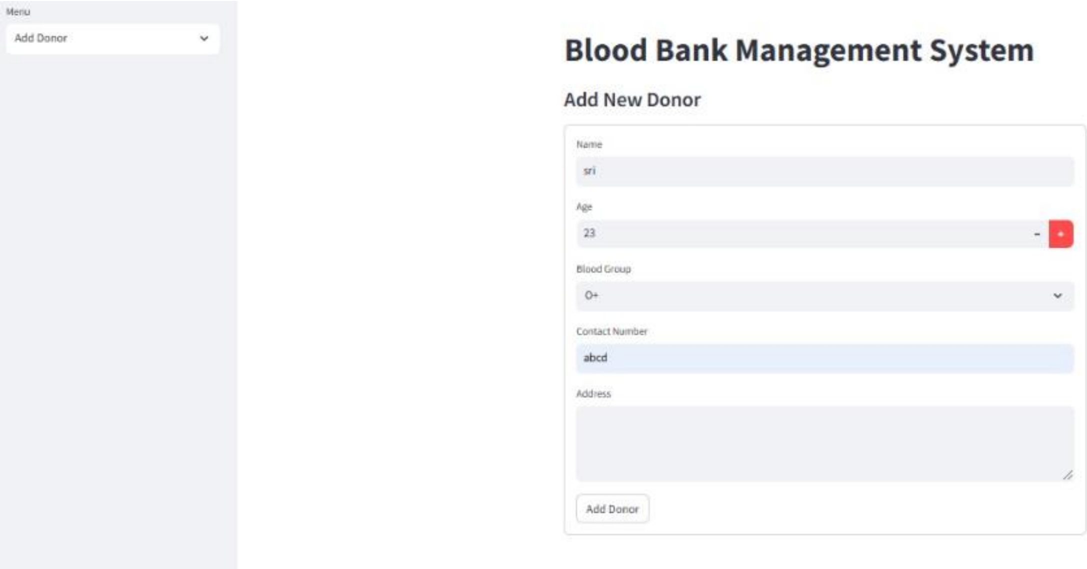


Blood Bank Management System

Welcome to the Blood Bank Management System!

0

Use the sidebar to navigate the system.



Blood Bank Management System

Add New Donor

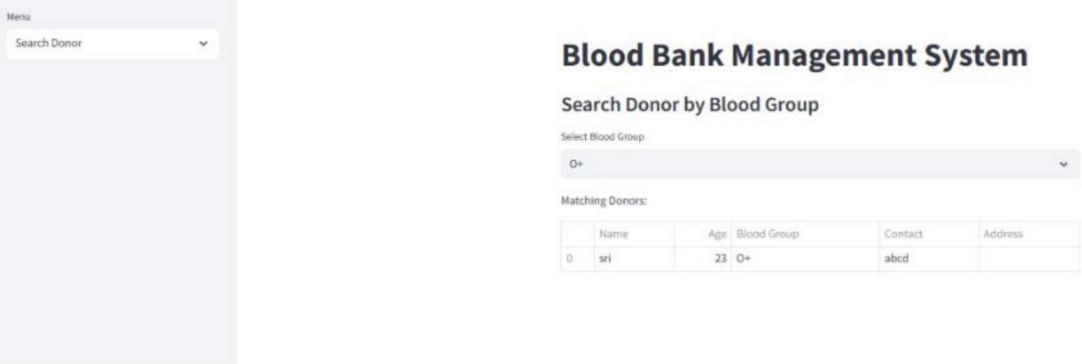
Name: sri

Age: 23

Blood Group: O+

Contact Number: abcd

Address:



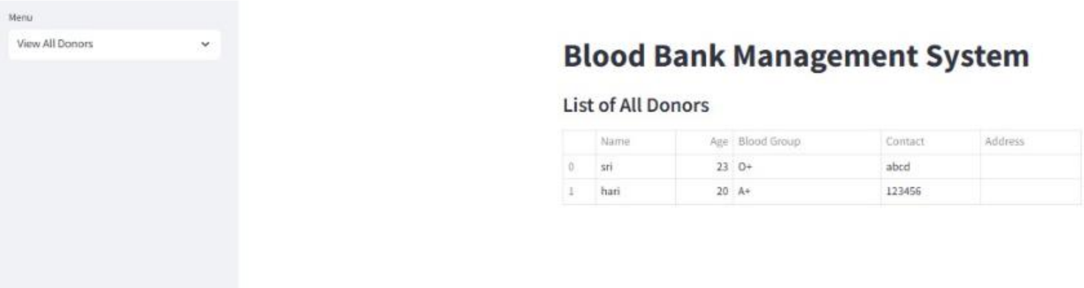
Blood Bank Management System

Search Donor by Blood Group

Select Blood Group: O+

Matching Donors:

	Name	Age	Blood Group	Contact	Address
0	sri	23	O+	abcd	

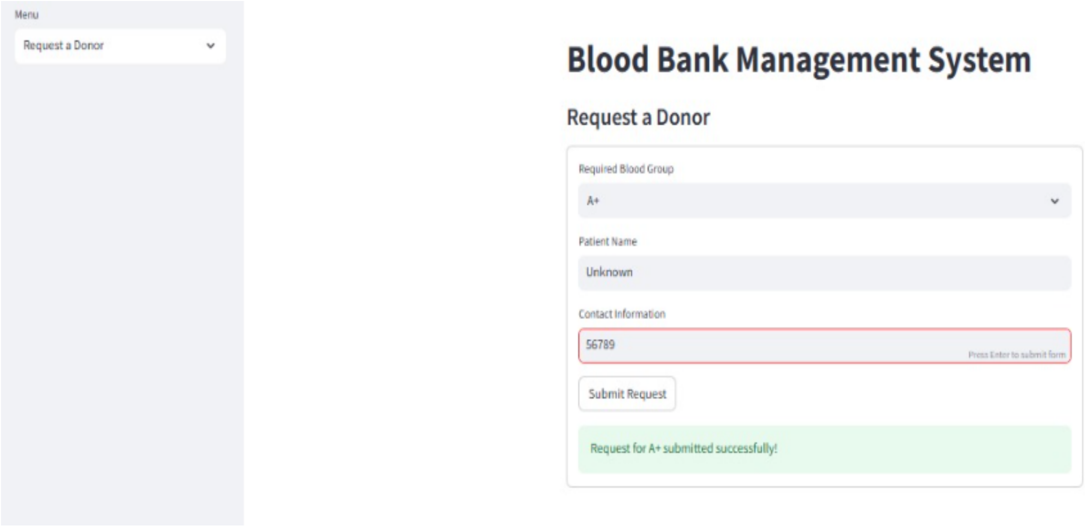


Blood Bank Management System

List of All Donors

	Name	Age	Blood Group	Contact	Address
0	sri	23	O+	abcd	
1	hari	20	A+	123456	

```
ELIF MENU == "ADD DONOR":

    ST.SUBHEADER("ADD NEW DONOR")

    WITH ST.FORM("DONOR_FORM"):

        NAME = ST.TEXT_INPUT("NAME")

        AGE = ST.NUMBER_INPUT("AGE", MIN_VALUE=18, MAX_VALUE=65, STEP=1)

        BLOOD_GROUP = ST.SELECTBOX("BLOOD GROUP", ["A+", "A-", "B+", "B-", "O+", "O-",
"AB+", "AB-"])

        CONTACT = ST.TEXT_INPUT("CONTACT NUMBER")

        ADDRESS = ST.TEXT_AREA("ADDRESS")

        SUBMITTED = ST.FORM_SUBMIT_BUTTON("ADD DONOR")



        IF SUBMITTED:

            ST.SESSION_STATE['DONORS'].APPEND({

                "NAME": NAME,

                "AGE": AGE,

                "BLOOD GROUP": BLOOD_GROUP,

                "CONTACT": CONTACT,

                "ADDRESS": ADDRESS

            })

            ST.SUCCESS(F"DONOR {NAME} ADDED SUCCESSFULLY!")



ELIF MENU == "SEARCH DONOR":

    ST.SUBHEADER("SEARCH DONOR BY BLOOD GROUP")

    BLOOD_GROUP_TO_SEARCH = ST.SELECTBOX("SELECT BLOOD GROUP", ["A+", "A-", "B+",
"B-", "O+", "O-", "AB+", "AB-"])

    FILTERED_DONORS = [DONOR FOR DONOR IN ST.SESSION_STATE['DONORS'] IF
DONOR['BLOOD GROUP'] == BLOOD_GROUP_TO_SEARCH]
```



Blood Bank Management System

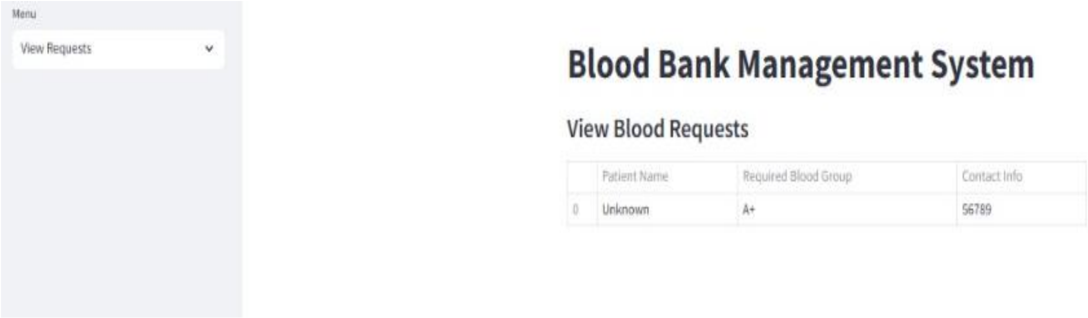
Request a Donor

Required Blood Group: A+

Patient Name: Unknown

Contact Information: 56789

Request for A+ submitted successfully!



Blood Bank Management System

View Blood Requests

	Patient Name	Required Blood Group	Contact Info
0	Unknown	A+	56789

```
IF FILTERED_DONORS:
        ST.WRITE("MATCHING DONORS:")
        DF = PD.DATAFRAME(FILTERED_DONORS)
        ST.TABLE(DF)
    ELSE:
        ST.WRITE("NO DONORS FOUND FOR THE SELECTED BLOOD GROUP.")


ELIF MENU == "VIEW ALL DONORS":
    ST.SUBHEADER("LIST OF ALL DONORS")
    IF ST.SESSION_STATE['DONORS']:
        DF = PD.DATAFRAME(ST.SESSION_STATE['DONORS'])
        ST.TABLE(DF)
    ELSE:
        ST.WRITE("NO DONORS HAVE BEEN ADDED YET.")


ELIF MENU == "REQUEST A DONOR":
    ST.SUBHEADER("REQUEST A DONOR")
    WITH ST.FORM("REQUEST_FORM"):
        BLOOD_GROUP_NEEDED = ST.SELECTBOX("REQUIRED BLOOD GROUP", ["A+", "A-",
"B+", "B-", "O+", "O-", "AB+", "AB-"])
        PATIENT_NAME = ST.TEXT_INPUT("PATIENT NAME")
        CONTACT_INFO = ST.TEXT_INPUT("CONTACT INFORMATION")
        SUBMITTED = ST.FORM_SUBMIT_BUTTON("SUBMIT REQUEST")


        IF SUBMITTED:
            ST.SESSION_STATE['REQUESTS'].APPEND({
                "PATIENT NAME": PATIENT_NAME,
                "REQUIRED BLOOD GROUP": BLOOD_GROUP_NEEDED,
```

```
            "CONTACT INFO": CONTACT_INFO

        })

        ST.SUCCESS(F"REQUEST FOR {BLOOD_GROUP_NEEDED} SUBMITTED
SUCCESSFULLY!")



ELIF MENU == "VIEW REQUESTS":

    ST.SUBHEADER("VIEW BLOOD REQUESTS")

    IF ST.SESSION_STATE['REQUESTS']:

        DF = PD.DATAFRAME(ST.SESSION_STATE['REQUESTS'])

        ST.TABLE(DF)

    ELSE:

        ST.WRITE("NO REQUESTS HAVE BEEN SUBMITTED YET.")
```

Conclusion:

The **Blood Bank Management System** developed using **Streamlit** offers a user-friendly interface for donors, health centre’s and administrators to efficiently manage blood donation data. This system centralizes key functionalities such as adding, updating, and viewing available blood records, tracking real time donation , and requesting for donors.