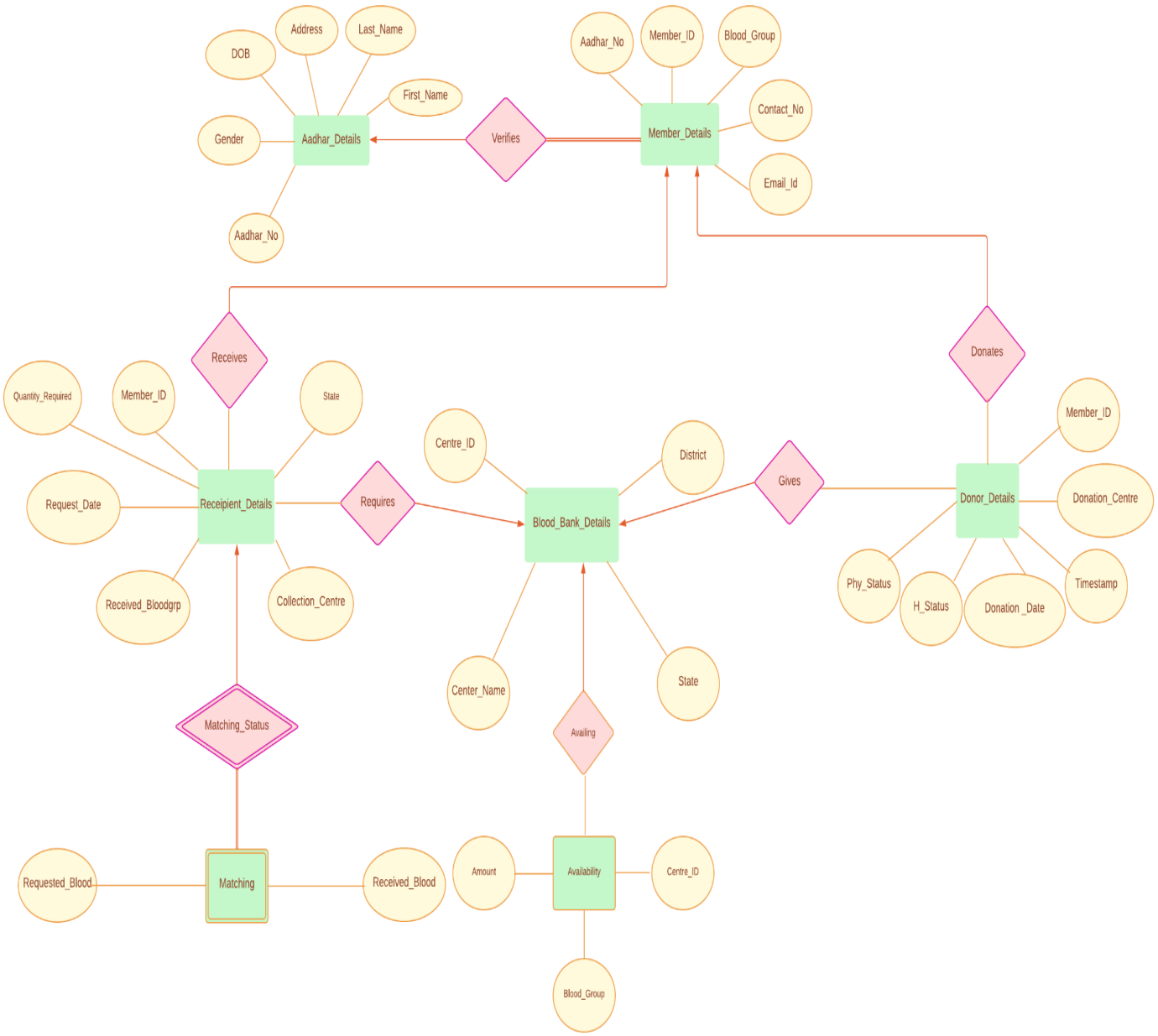
BLOODBANK MANAGEMENT SYSTEM DOCUMENTATION

ER Diagram using Lucidchart & the relationship between the entities



INFORMATION ABOUT OUR ENTITIES

We have 7 entities the description of which are stated as under:

Aadhar_Details:(Attributes-Aadhar_No,First_Name,Last_Name, Address,DOB,Gender)
This entity holds information on all people who may one day become members of the blood bank management system. In other words, it comprises people's aadhar records, which may have been obtained from official databases.Here, Aadhar_No acts as our primary key.

Member_Details:(Attributes-Aadhar_No, Member_ID, Blood_Grp,Email,Contact)
This entity holds information on everyone who has registered as a member of the blood bank system. It is not mandatory for a member to give or receive blood at least once before becoming a member.Primary key is Member_ID and Foreign key is Aadhar_No.

Receipient_Details:(Attributes- Member_ID, Quantity_Required,Request_Date,Received_Bloodgrp, Collection_Centre,State)
This entity holds the data of all members who have received blood to the blood bank, as well as information on their receiving centres.Primary keys:- Member_ID,Request_Date.Foreign key: Member_ID

Blood_Bank_Details:(Attributes: Centre_ID, Centre_Name,State,District)
This entity contains the data of all the blood centers where one can receive or donate blood.Each center contains a unique centre_Id.Primary key-Centre_ID

Donor_Details: (Attributes: Member_ID, Donation_Centre,Timestamp,Donation_Date,H_Status,Phy_Status)
This entity includes the data of all blood bank members who have donated blood at least once to the blood bank. A member may give blood if and only if they have a healthy hemoglobin level and are physically fit.Primary Key-Member_ID, Donation_Centre Candidate Keys-Member_ID, Donation_Date

Matching:(Attributes:Requested_Blood, Received_Blood)
This entity mostly contains information that diminishes the compatibility of distinct blood types with one another. It is a

vulnerable entity that relies on the Receipient_Details to uniquely identify itself.

Availability:(Attributes: Amount,Centre_Id,Blood_Grp)
This entity shows the availability status of different blood types at different centres.Primary Key- Centre_Id,Blood_Grp.Foreign Key-Centre_Id.

RELATIONSHIP BETWEEN ENTITIES

Aadhar_Details and Member_Details
Relationship:Verifies
Type of relation:One to one with total participation of Member_Details

Member_Details and Donor_Details
Relationship:Donates
Type of relation:One to many

Member_Details and Receipient_Details
Relationship:Receives
Type of Relation:One to many relationship

Receipient_Details and Blood_Bank_Details
Relationship:Requires
Type of Relation:Many to one

Donor_Details and Blood_Bank_Details
Relationship:Gives
Type of Relation:Many to one

Availability and Blood_Bank_Details
Relationship:Availing
Type Of Relation:Many to one

Receipient_Details and Matching
Relationship:Matching_Status
Weak Entity
Type of Relation:one to many with total participation of weak entity

TABLES CREATED IN OUR DATABASE

The Aadhar_Details Table

The Availability Table

The Donor_Details Table

	Member_ID	Donation_Centre	timsetamp	Donation_Date	Haemoglobin_Status	Physical_Status
*	NULL	NULL	NULL	NULL	NULL	NULL

The Blood_Bank_Centre

	Centre_ID	Centre_Name	District	State
*	NULL	NULL	NULL	NULL

The Matching Table

	Requested_Blood	Received_Blood

The Member_Details Table

	Aadhar_No	Member_ID	Blood_Group	Contact_No	Email_ID
*	NULL	NULL	NULL	NULL	NULL

The Receipient_Details Table

	Member_ID	Collection_Centre	State	Quantity_Required	Request_Date	Received_Blood_Group
*	NULL	NULL	NULL	NULL	NULL	NULL

Normalization Of Blood Bank Database

For Aadhar_Details

Aadhar_No => Aadhar_No(functional dependency exists, because two different Aadhar_No
do not correspond to the same Aadhar_No)
Aadhar_No =>First_Name(functional dependency exists).
Aadhar_No => Last_Name(functional dependency exists).
Aadhar_No =>DOB(functional dependency exists).
Aadhar_No =>Gender(functional dependency exists).

For Member_Details

Member_ID =>Member_ID(functional dependency exists, because two different Member_ID
do not correspond to the same Member_Id)
Member_ID =>Aadhar_No(functional dependency exists)
Member_ID =>Blood_grp(functional dependency exists)
Member_ID =>Contact_No(functional dependency exists)
Member_ID =>Email(functional dependency exists)

For Blood_Bank_Centre

Centre_ID =>Centre_ID(functional dependency exists, because two different Centre_ID
do not correspond to the same Centre_Id)
Centre_ID =>Centre_Name(functional dependency exists)
Centre_ID =>District(functional dependency exists)
Centre_ID =>State(functional dependency exists)

For Availability

Centre_ID,Blood_grp =>Centre_ID(Trivial functional dependency exists)
Centre_ID,Blood_grp =>Blood_grp(Trivial functional dependency exists)

Centre_ID,Blood_grp =>Amount(functional dependency exists)

For Donor_Details

Member_ID,Donation_Centre =>Member_ID(Trivial functional dependency exists)

Member_ID,Donation_Centre =>Donation_Centre(Trivial functional dependency exists)

Member_ID,Donation_Centre =>Timestamp(functional dependency exists)

Member_ID,Donation_Centre =>Donation_Date(functional dependency exists)

Member_ID,Donation_Centre =>Haemoglobin_Status(functional dependency exists)

Member_ID,Donation_Centre =>Physical_Status(functional dependency exists)

Similarly, we have another set of candidate key
Member_Id,Donation_Date

For Receipient_Details

Member_ID, Request_Date =>Member_ID(Trivial functional dependency exists)
Member_ID,Request_Date =>Request_Date(Trivial functional dependency exists)

Member_ID,Request_Date =>Collection_Centre(functional dependency exists)

Member_ID,Request_Date =>State(functional dependency exists)

Member_ID,Request_Date =>Quantity_Required(functional dependency exists)

Member_ID,Request_Date =>Received_Bloodgrp(functional dependency exists)

Thus, all of our tables are in 1st, 2nd, 3rd, and Boyce-Codd Normal Form.

Implementation Of Insert Procedures

The following sections contain the list of procedures and their implementation

1)**check_member**

It verifies if the credentials entered by the member are legitimate by comparing them to the aadhar information.

2) **insert_member**

This procedure inserts values into the blood management system.

```
CALL insert_member('7894785678127823','B+ve','8837365354','farah@gmail.com');
CALL insert_member('7894785678127824','A+ve','7894561230','ruchika@hotmail.com');
CALL insert_member('7894785678127825','AB+ve','9456874236','sanam@gmail.com');
```

Aadhar_No	Member_ID	Blood_Group	Contact_No	Email_ID
7894785678127823	4	B+ve	8837365354	farah@gmail.com
7894785678127824	5	A+ve	7894561230	ruchika@hotmail.com
7894785678127825	6	AB+ve	9456874236	sanam@gmail.com
NULL	NULL	NULL	NULL	NULL

3) **insert_donor**

This procedure inserts donor details who want to donate blood. It also ensures that the donor is a part of the member database.

```
CALL insert_donor(4, 'NL01', '2022-01-01');
CALL insert_donor(5, 'RJ02', '2022-04-20');
CALL insert_donor(6, 'RJ02', '2022-05-01');
```

Member_ID	Donation_Centre	timsetamp	Donation_Date	Haemoglobin_Status	Physical_Status
4	NL01	2022-04-13 18:27:14	2022-01-01	NULL	0
5	RJ02	2022-04-13 18:27:14	2022-04-20	NULL	0
6	RJ02	2022-04-13 18:27:14	2022-05-01	NULL	0
NULL	NULL	NULL	NULL	NULL	NULL

4)**insert_centre**

Procedure inserts new centre into the blood bank database.
It calls the ‘insert_avail’ procedure to insert rows associated with the new inserted centre.

```
CALL insert_centre('NL01','Dimapur Blood Centre','Dimapur','Nagaland');
CALL insert_centre('RJ02','Bhilwara Blood Centre','Bhilwara','Rajasthan');
CALL insert_centre('RJ01','Pilani Blood Centre','Pilani','Rajasthan');
```

Centre_ID	Centre_Name	District	State
NL01	Dimapur Blood Centre	Dimapur	Nagaland
RJ01	Pilani Blood Centre	Pilani	Rajasthan
RJ02	Bhilwara Blood Centre	Bhilwara	Rajasthan
NULL	NULL	NULL	NULL

5)**insert_avail**

This procedure is called by ‘insert_centre’ when a new centre is inserted inorder to fill in the blood details in the ‘availability’ table.

6)**find_match**

The procedure takes Member ID, State of the Recipient and the quantity of blood required as inputs. It returns the list of all possible centre such that the centre is the output which lies in the state given as an input by the user.Matching of blood groups is

done on the basis of a matching table.It follows the real life blood matching rather than one to one similar matching.
For example, If A+ is given as input by the user it returns all those centres from which the user can avail not only the matching blood A+ but also the other compatible blood groups like O+,O-,A-

```
CALL find_match(6,'Rajasthan', 200);
```

Centre_Name
Bhilwara Blood Centre

7)**select_centre**

It allows the user to select the favorable centre from the list of centres by ‘find_match’

```
CALL select_centre('Bhilwara Blood Centre',6, 'Rajasthan', '2022-04-13', 200);
```

Member_ID	Collection_Centre	State	Quantity_Required	Request_Date	Received_Blood_Group
6	RJ02	Rajasthan	200	2022-04-13	A+ve
NULL	NULL	NULL	NULL	NULL	NULL

Centre_ID	Blood_Group	Amount
RJ02	A-ve	0
RJ02	A+ve	100

8)**fetch_blood_grp**

This procedure returns the blood group of a particular member.

9)**update_donor**

It updates the donor details after checking for the haemoglobin_status and fitness status.

```
CALL update_donor(4,'2022-01-01');
CALL update_donor(5,'2022-04-20');
```

Centre_ID	Blood_Group	Amount
NL01	B-ve	0
NL01	B+ve	300
NL01	O-ve	0
NL01	O+ve	0
RJ01	A-ve	0
RJ01	A+ve	0
RJ01	AB-ve	0
RJ01	AB+ve	0
RJ01	B-ve	0
RJ01	B+ve	0
RJ01	O-ve	0
RJ01	O+ve	0
RJ02	A-ve	0
RJ02	A+ve	300

10)**update_availability**

Increments or decrements the quantity of available blood depending upon the functions carried out.

There are more procedures that are self-explained in the SQL file.

BETTER PROSPECTS IN LATER VERSIONS

- Triggers can be used before inserting details into tables
- Blood can be availed not only from one state but also from the neighboring states based on the convenience of distance in future versions.
- The concept of Member_ID could be enhanced by implementing a hashing function for generating random unique ID rather than just auto-incrementing.
- Our project just provides one-way interaction. In future versions, we can focus on publicizing it in order to make it more interactive and allow multi-user use it.