Chapter 1:

INTRODUCTION

1.1 OVERVIEW

Databases store data and metadata. Data are the individual facts that are used to derive information. Metadata describe the content, quality, condition, availability, and characteristics of data. Database Management Systems (DBMS) is used to modify the data. There are varieties of database types: Sequential files, Hierarchical databases, Network databases, and Relational databases. The recent database type is Object-Relational database, which is essentially a relational database with some Object properties. Relational databases become popular because it was easy to modify the schema. It is very easy to add tables and columns to the schema, and doing so does not affect the remainder of the schema, and more important, does not affect the applications that access the database schema. Older databases required the databases to be restructured and the applications to be modified. Avoiding database and application maintenance is an important benefit and the reason for the switch. Relational databases consist of independent tables. Relational Database Management Software (RDBMS) does not know how the records in the table are related.

In the realm of healthcare, the efficient management of blood resources is paramount for saving lives and ensuring the well-being of patients. The Blood Bank Management System (BDMS) stands as a technological cornerstone, addressing the intricate challenges associated with the collection, testing, storage, and distribution of blood and its components.

At its core, BDMS is a comprehensive software solution designed to streamline and enhance the entire blood management process. This system serves as the linchpin connecting donors, healthcare professionals, and patients, offering a sophisticated platform that integrates various functionalities to ensure the seamless functioning of blood banks. One of the primary facets of BDMS is donor management, encompassing the registration, screening, and maintenance of donor records. The system facilitates the meticulous tracking of donor histories, ensuring a readily available pool of safe and compatible blood units. Through secure authentication processes, the BDMS ensures the reliability of donor information and promotes a culture of regular blood donations.

Furthermore, BDMS tackles the complex logistics of blood inventory management. It provides real-time monitoring of blood stocks, expiration dates, and ensures the efficient utilization of available resources. The system enables healthcare institutions to respond promptly to emergencies, guaranteeing the swift availability of blood units tailored to patients' needs. The integration of various modules within BDMS, such as Admin, Donors, Blood Inventory, Donation Records, and Transfusion History, allows for a holistic approach to blood bank operations.

Administrators benefit from functionalities like user authentication, blood inventory tracking, and the ability to manage donor and transfusion records seamlessly. Simultaneously, donors experience a user-friendly interface for registration, appointment scheduling, and accessing their donation history. As a testament to its efficacy, BDMS plays a pivotal role in maintaining data integrity through interdependence among different database tables. Dynamic updates to records, such as Donation Records reflecting real-time blood donations and transfusions, underscore the system's commitment to accuracy and efficiency. In conclusion, the Blood Bank Management System emerges as an indispensable tool in the healthcare landscape. Its sophisticated design, user-friendly interface, and robust functionalities contribute to the optimization of blood bank operations. BDMS not only ensures the availability of life-saving blood units but also fosters a sustainable and organized approach to blood donation, underscoring its significance in promoting public health and well-being

1.2 ORACLE

Oracle is the largest database manufacturer and the second-largest software manufacturer in the world. The company began as a relational database manufacturer. In the beginning, Oracle touted its software as “being able to run on any platform”. This openness has been most attractive to companies; an Oracle has tried to maintain its image as an open product. Oracle was at a good place when industry became extremely interested in moving away from network databases and the mainframe.

Oracle Database 10g is an entry-level database based on the Oracle Database 10g Release 2 code base that's free to develop, deploy, and distribute; fast to download; and simple to administer. Oracle Database is a great starter database for:

- Developers working on PHP, Java, .NET, and Open Source applications
- DBAs who need a starter database for training and deployment
- Independent Software Vendors (ISVs) and hardware vendors who want a starter database
- Educational institutions and students who need database for their curriculum

1.3 MYSQL

MySQL is an open-source relational database management system widely used for building and managing databases. With cross-platform compatibility, it supports SQL for data definition and manipulation. MySQL's client-server architecture enables efficient data management, and its security features include user authentication and access control. Known for high performance and scalability,

MySQL supports various storage engines such as InnoDB. A robust community provides ample support and resources. Widely used in web development, MySQL is a key component in stacks like LAMP. It offers a commercial enterprise edition with additional features for advanced requirements.

Some key points about MySQL are :-

- **Open Source**: MySQL is an open-source relational database management system, allowing users to view, modify, and distribute its source code freely.
- **RDBMS**: MySQL follows the principles of a Relational Database Management System, organizing data into tables with rows and columns.
- **Cross-Platform Compa**tibility: MySQL is cross-platform and can run on various operating systems, including Linux, Windows, and macOS.
- **Client-Server Architecture**: MySQL employs a client-server architecture, allowing multiple clients to connect to a central server to execute SQL queries and commands.
- **Web Development**: MySQL is widely used in web development, particularly in conjunction with popular web development stacks like LAMP (Linux, Apache, MySQL, PHP/Python/Perl).

1.4 BASICS

Structured Query Language (SQL), which is an ANSI standard language for interacting with relational databases, is the main tool for extracting the information.

A **database** is a representation of a real-world thing called an **Entity.** Examples of entities are vehicles, employees, customers, fish, buildings, and even things such as baseball teams. The database stores facts about the entity in an organized framework, model, or schema. These facts are called **attributes.**

An **Instance** is one occurrence of an entity. Each entity must have an identifier, which is one or more attributes that make each entity instance unique from any other instance. The identifier should contain a value that does not change. Examples of identifiers are student IDs, payroll numbers, or social security numbers.

If the entity does not have an attribute that can be used as an identifier, an artificial identifier can be created. The identifier on an entity is often called a **primary key.**

A **foreign key** is a set of attributes of the considered table that exists as a primary key attributes in another table. Database records are matched (joined) through the use of primary and foreign keys.

Normalization is a process consisting of series of steps, which is used to group the database attributes. The purpose of this design is to ensure that the tables within the database are space efficient and performance efficient.

- Zero Normal Form—each of the relations (tables) has a unique identifier (primary key).
- First Normal Form—Separate the repeating groups of attributes or multi-valued attributes into a relation of their own. Be sure to form composite keys.
- Second Normal Form—Establish full functional dependency by separating out attributes that are not fully dependent on the full primary keys.
- Third Normal Form—Remove transitive dependencies by separating attributes that are dependent on a non key attribute.

Chapter 2:

REQUIREMENT SPECIFICATION

2.1 Hardware Requirements

- Operating System: -windows 10
- Processor: - i5
- Processor Speed: - 3.4 GHZ
- Installed Memory (RAM) :- 8 GB
- Hard Disk :- 500 GB
- System Type: - 64-bit OS

2.2 Software Requirements

- Front End :- JSP/CSS/BOOTSTRAP/JS
- Back End :- SERVLET/JDBC/JAVA/MYSQL
- Server :- APACHE TOMCAT 10.1 or greater
- JDK 19 or more
- MYSQL Connector

2.3 Tools and Technology used.

Tools :

- **JSP [Java Server Page]** :-

  Java Server Pages (JSP) is a technology that enables the development of dynamic, server-side web applications in Java. JSP pages consist of HTML or XML content combined with Java code snippets, facilitating the creation of dynamic content. JSP simplifies the integration of Java code into web pages, allowing developers to embed Java directly within HTML. By leveraging reusable components called custom tags, JSP promotes modular and maintainable code. These pages are executed on the server, generating dynamic content that is then sent to the client's web browser. JSP is commonly used in conjunction with servlets to build robust and scalable Java web applications.

- **CSS [Cascading Style Sheet]** :-

  CSS (Cascading Style Sheets) is a web styling language used to design and format HTML documents. It employs selectors to target HTML elements and define styles such as colors, fonts, and layouts. Following the box model, it structures elements with content, padding, border, and margin. CSS promotes a separation of concerns by isolating styling from HTML content for better maintainability. It enables responsive design through media queries and supports a hierarchy of selectors for specific rule application.

- **JS [Java Script]** ;-

  JavaScript is a versatile programming language primarily used for client-side web development, enabling dynamic and interactive features in web pages. It follows an event-driven paradigm, responding to user actions like clicks and input. JavaScript is also utilized for server-side development through platforms like Node.js, making it a full-stack language. Its asynchronous nature allows non-blocking execution, crucial for tasks such as AJAX requests. JavaScript is object-oriented, utilizing prototypes for inheritance, fostering code reusability.

- **Servlet** :-

  Servlets are Java-based server-side components used in web development. They handle HTTP requests, allowing dynamic content generation within web applications. With a defined lifecycle including init(), service(), and destroy() methods, servlets provide control over initialization, request processing, and cleanup. Platform-independent, servlets run on servers supporting the Java Servlet API.

- **JDBC [Java Database Connectivity]** :-

  Java Database Connectivity (JDBC) is a Java-based API that facilitates communication between Java applications and relational databases.

  Connectivity steps are : -

  1. Loading JDBC Driver:
  2. Creating Database Connection:
  3. Creating Statement:
  4. Executing SQL Queries:
  5. Handling Results:
  6. Closing Resources:

Chapter 3:

DESCRIPTION

3.1 ER Diagram

An entity-relationship model (ER model) is a data model for describing the data or information aspects of a business domain or its process requirements, in an abstract way that lends itself to ultimately begin implemented in the database such as a relational database. The main components of ER model are entities (things) and the relationship that can exist among them. Entity – relationship modeling was developed by Peter Chen and published in 1976 paper. The ER diagram is drawn to have a better understanding of the whole scenario, it is used to conceptualize the phenomena, actions, and interactions between various entities and to arrive at the specific requirements in a comprehensive manner.

An entity relation relationship model is the result of using a systematic process to describe and define a subject area of business data. The data is represented as components (entities) that are linked with each other by relationships that express the dependencies and requirements between them, such as: one building may be divided into number of apartments, but one apartment can be located in one building. Entities may have various properties (attributes) that characterize them.

Diagrams created to represent these entities, attributes, and relationships graphically are called entity-relationship diagrams. An ER model is typically implemented as database. In the case of a relational database, which stores data in tables, every row of each table represents one instance of an entity. Some data fields in these tables to indexes in other tables; such pointers are physical implementation of the relationships.
The three- schema approach to software engineering uses three levels of ER models may be developed.

A relationship captures how entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns. Entities and relationships can both have attributes. The ER diagram for this database is as shown

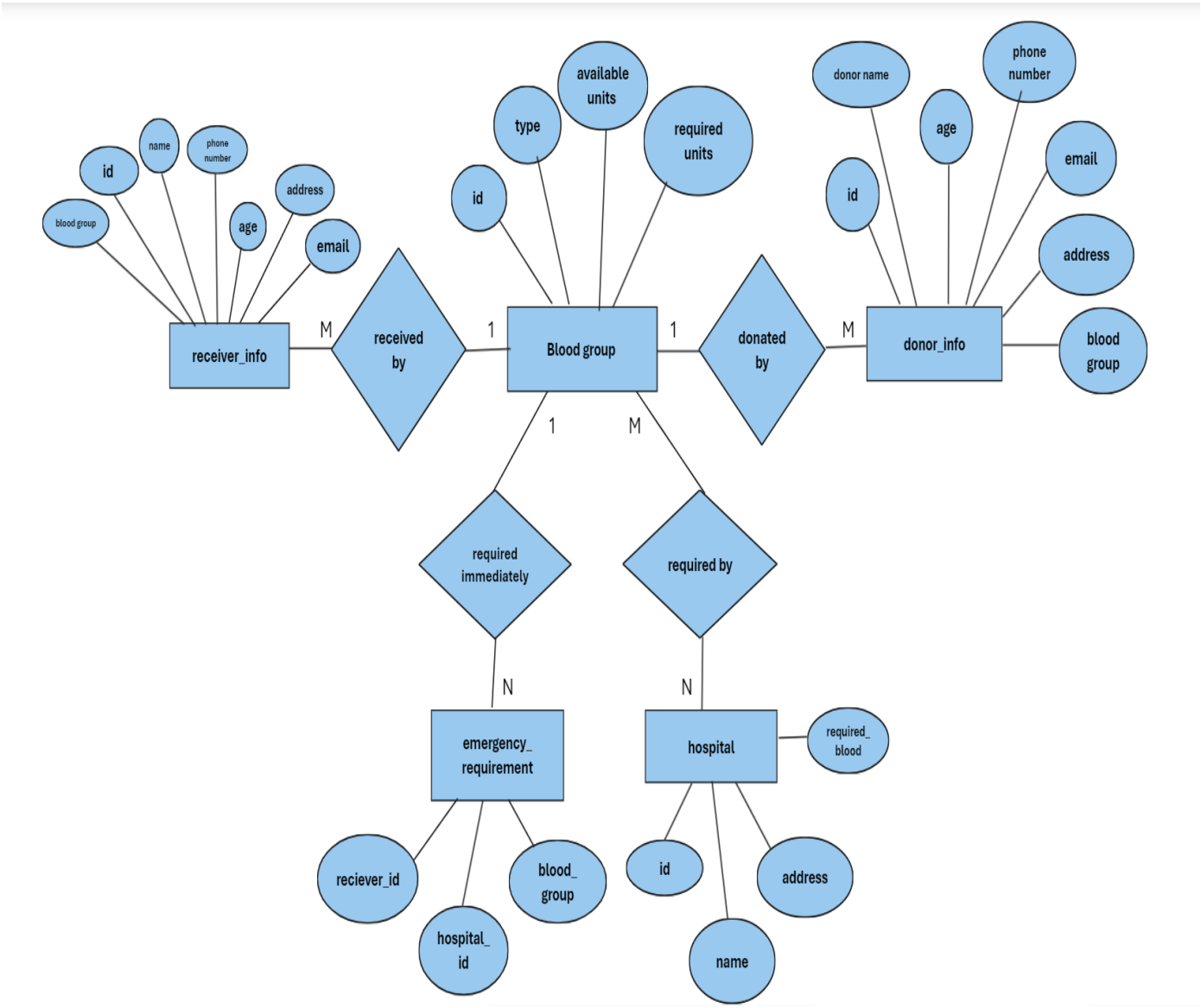
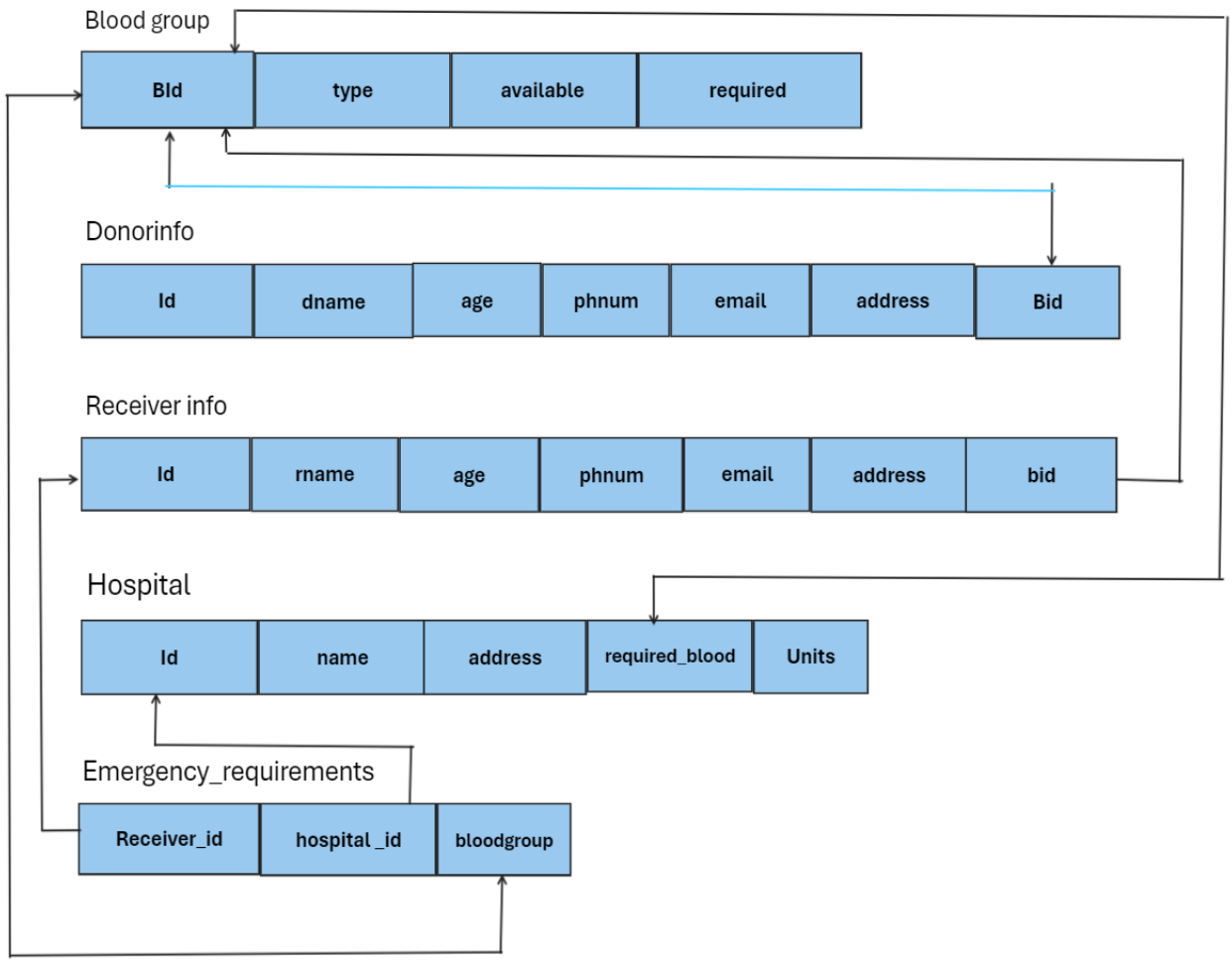


Fig.3.1-ER Diagram of Bood bank Management System

Fig.3.2-Schema Diagram



Chapter 4:

CODING

4.1 Table Creation

- **USERS**

```
CREATE TABLE USERS(
ID INT PRIMARY KEY AUTO_INCREMENT,
USER_NAME VARCHAR(20),
USER_PASSWORD VARCHAR(20));
```

- **BLOOD GROUPS**

```
CREATE TABLE BLOOD_GROUPS(
ID INT PRIMARY KEY ,
BLOOD_TYPE VARCHAR(3),
IN_STOCK INT,
REQUIRED_UNITS INT);
```

- **DONOR INFO**

```
CREATE TABLE DONORINFO(
ID INT PRIMARY KEY AUTO_INCREMENT,
DNAME VARCHAR(30),
AGE INT ,
PHNUM VARCHAR(10),
EAMIL VARCHAR(30),
ADDRESS VARCHAR(50),
BLOODGROUP INT,
FOREIGN KEY(BLOODGROUP) REFERENCES BLOOD_GROUPS(ID));
```

- **RECEIVER INFO**

```
CREATE TABLE RECEIVERINFO(
ID INT PRIMARY KEY AUTO_INCREMENT,
RNAME VARCHAR(30),
AGE INT ,
PHNUM VARCHAR(10),
EMAIL VARCHAR(30),
ADDRESS VARCHAR(50),
BLOODGROUP INT,
FOREIGN KEY(BLOODGROUP) REFERENCES BLOOD_GROUPS(ID));
```

- **EMERGENCY_REQUIREMENTS**

```
CREATE TABLE EMERGENCY_REQUIREMENT(
RECEIVER_ID INT ,
HOSPITAL_ID INT,
BG INT,
FOREIGN KEY(RECEIVER_ID) REFERENCES RECEIVERINFO(ID),
FOREIGN KEY(HOSPITAL_ID) REFERENCES HOSPITAL(ID),
FOREIGN KEY(BG) REFERENCES BLOOD_GROUPS(ID));
```

- **HOSPITALS**

```
CREATE TABLE HOSPITAL(
ID INT PRIMARY KEY,
HOSP_NAME VARCHAR(50),
HOSP_ADDRESS VARCHAR(50),
REQUIRED_TYPE INT,
REQUIRED_UNITS INT,
FOREIGN KEY(REQUIRED_TYPE) REFERENCES BLOOD_GROUPS(ID)
);
```

QUERIES

1.Inserting data into donorinfo:

```
INSERT INTO DONORINFO(
DNAME,AGE,PHNUM,EMAIL,ADDRESS,BLOODGROUP)
VALUES(?,?,?,?,?,?);
UPDATE BLOODGROUP
SET IN_STOCK = IN_STOCK + 1
WHERE  BLOOD_TYPE=?;
```

2.Inserting data into receiverinfo:

```
INSERT INTO RECEIVERINFO(
RNAME,AGE,PHNUM,EMAIL,ADDRESS,BLOODGROUP)
VALUES(?,?,?,?,?,?);
UPDATE BLOODGROUP
SET REQUIRED_UNITS = REQUIRED_UNITS + 1
WHERE  BLOOD_TYPE=?;
```

3.Creating hospital VIEW and displaying it :

```
CREATE  VIEW  HOSPITALVIEW AS
SELECT HOSPITAL.ID,HOSP_NAME,HOSP_ADDRESS,BLOOD_TYPE,HOSPITAL.REQUIRED_UNITS
FROM HOSPITAL ,BLOOD_GROUPS
WHERE BLOOD_GROUPS.ID=HOSPITAL.REQUIRED_TYPE ;
SELECT * FROM HOSPITALVIEW;
```

4.Login :

```
SELECT ID
FROM USERS
WHERE USER_NAME='?' AND USER_PASSWORD='?';
```

5.Display donor info:

```
SELECT ID,AGE,DNAME,ADDRESS,BLOODGROUP
FROM DONORINFO
ORDER BY DNAME;
```

6.Display reciever info:

```
SELECT ID,AGE,RNAME,ADDRESS,BLOODGROUP
FROM DONORINFO
ORDER BY RNAME;
```

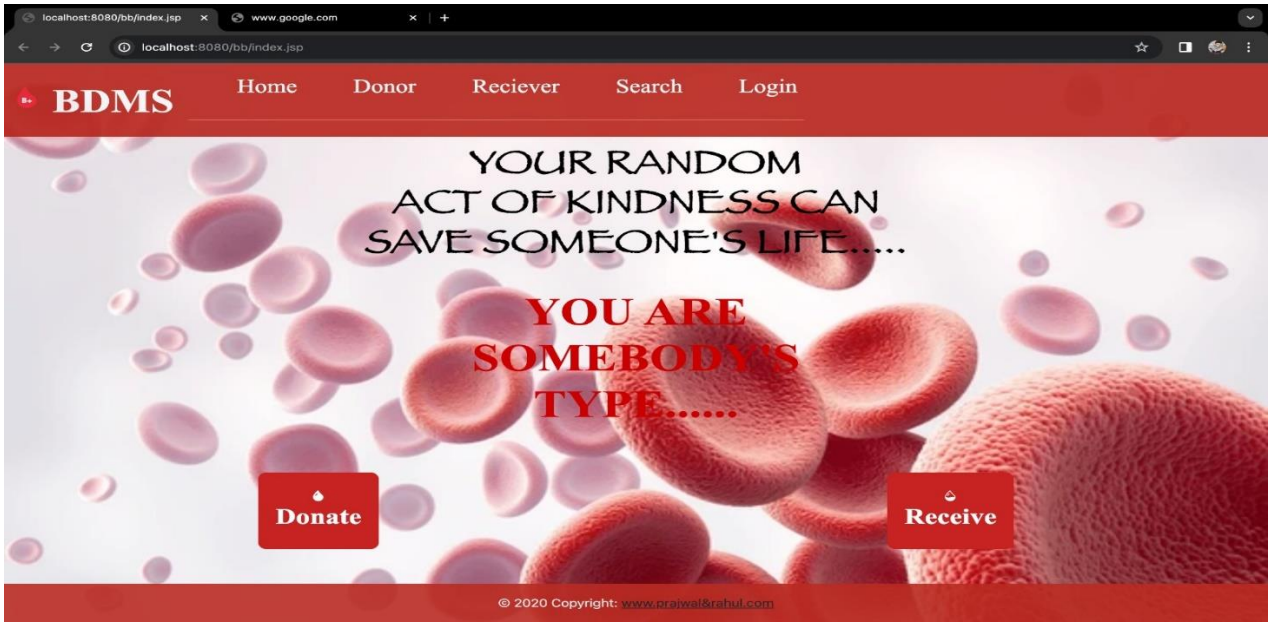
7.Search required blood:

```
SELECT ID,AGE,DNAME,ADDRESS,BLOODGROUP
FROM DONORINFO
WHERE BLOODGROUP="?”AND ADDRESS="?"
ORDER BY BLOODGROUP;
```

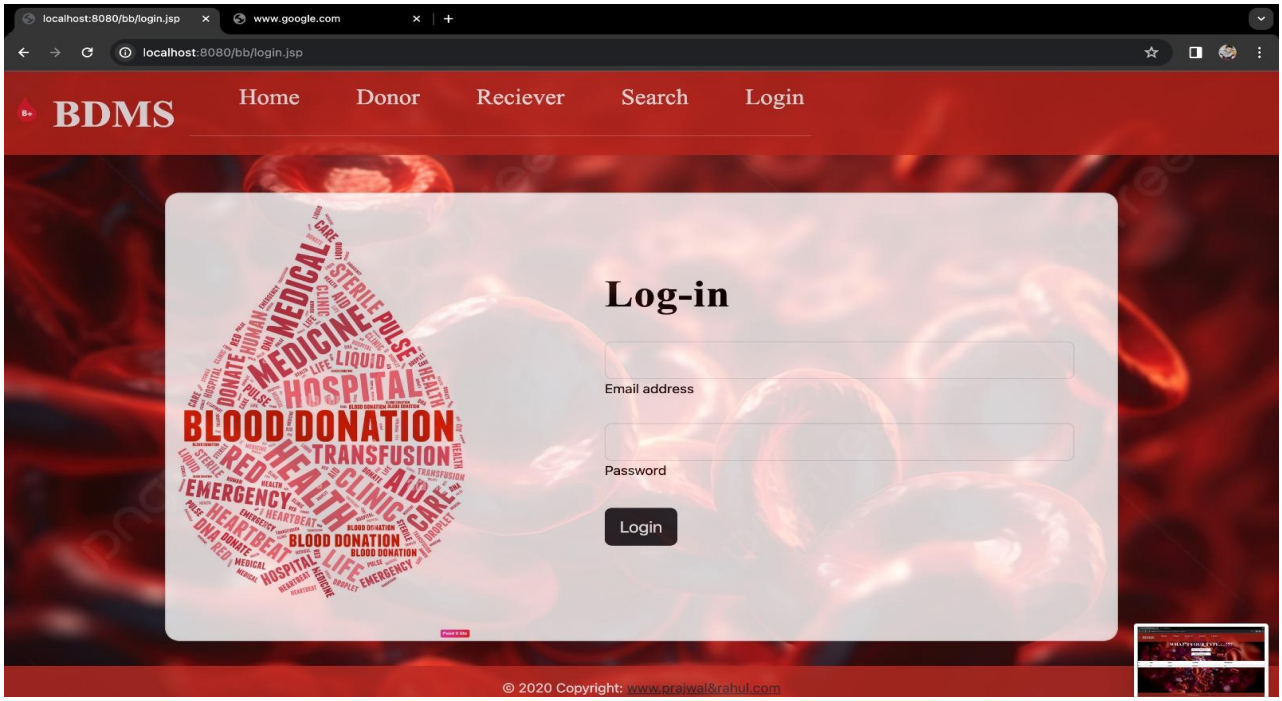
Chapter 5:

SNAPSHOTS

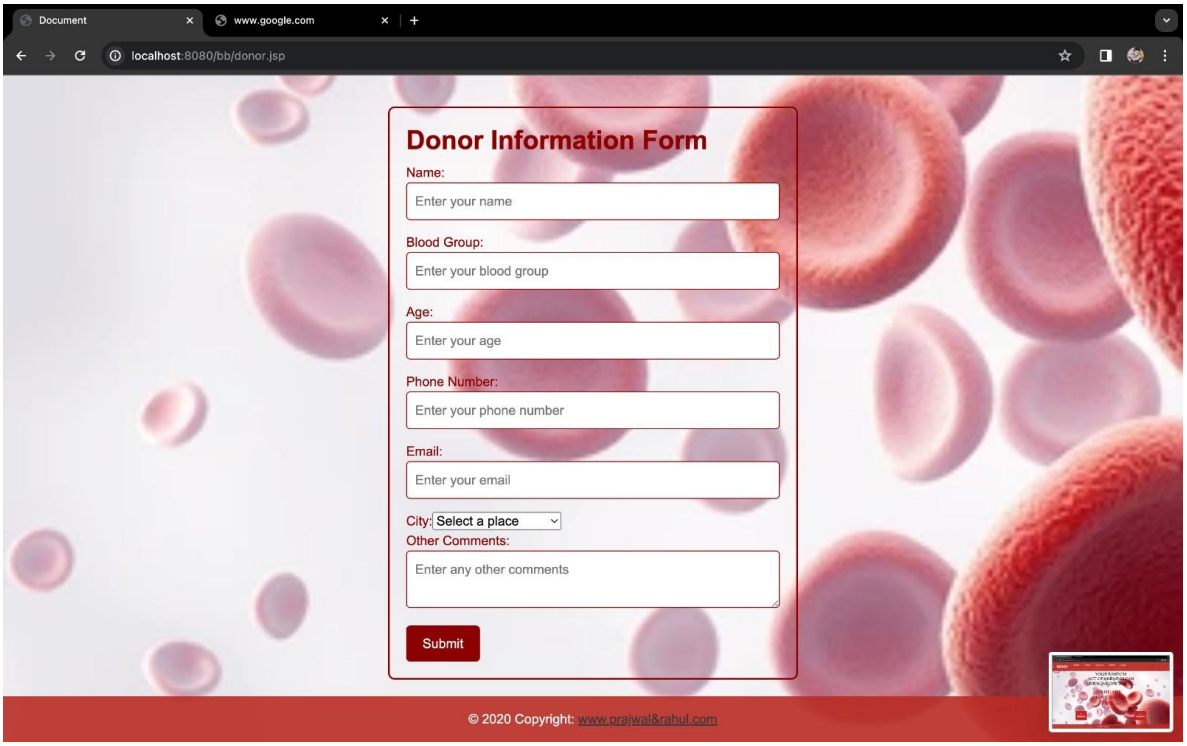
1. HOME PAGE



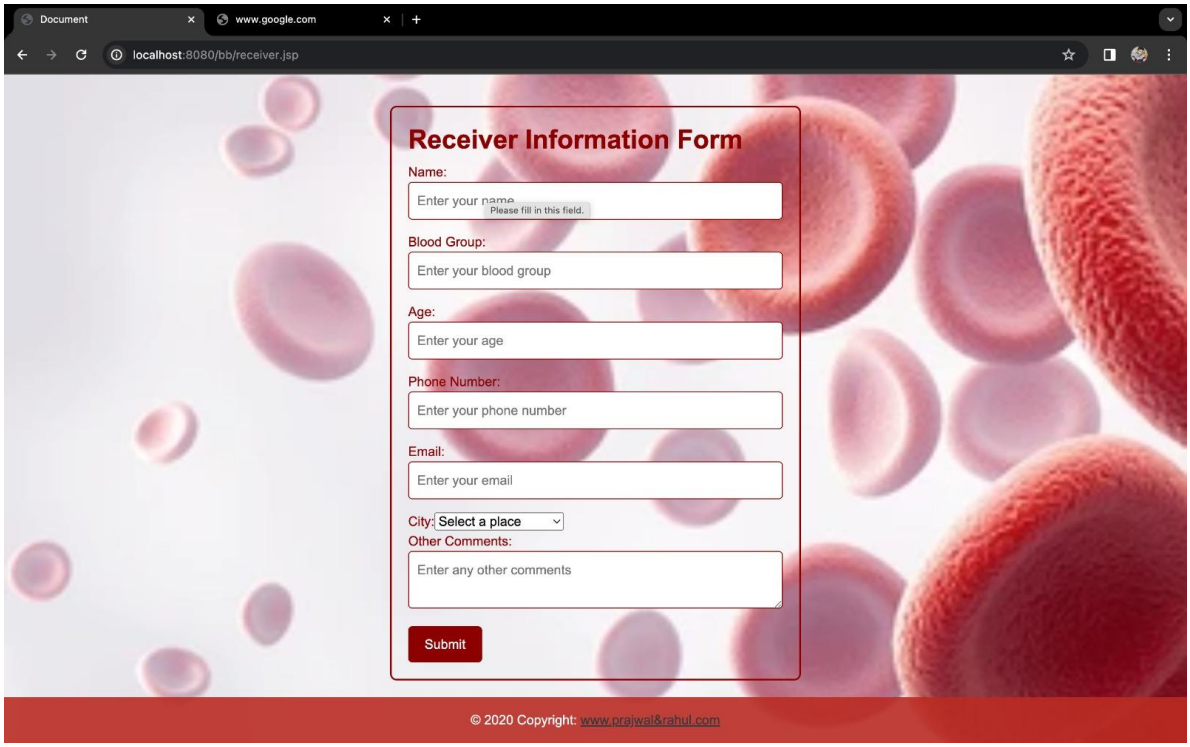
2. ADMIN LOGIN



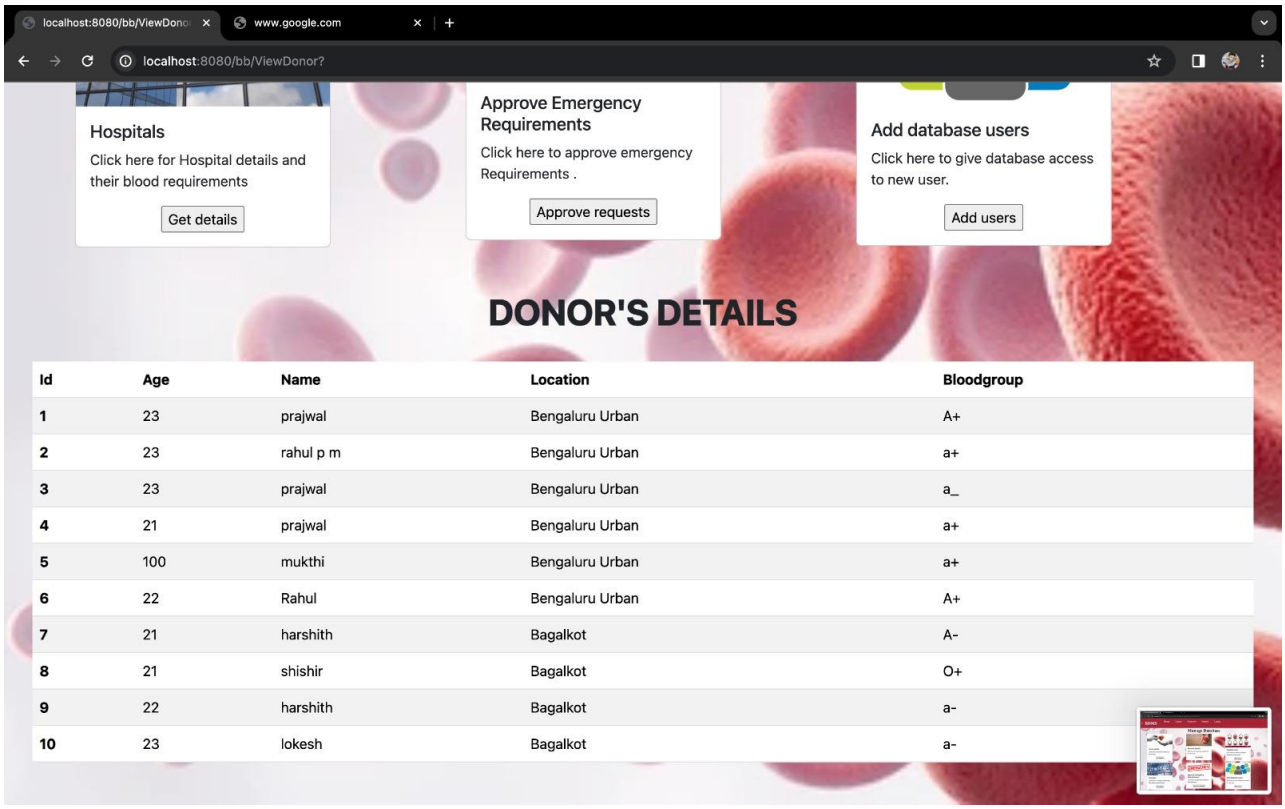
3. DONOR INFO- FORM



4. RECEIVER INFO-FORM

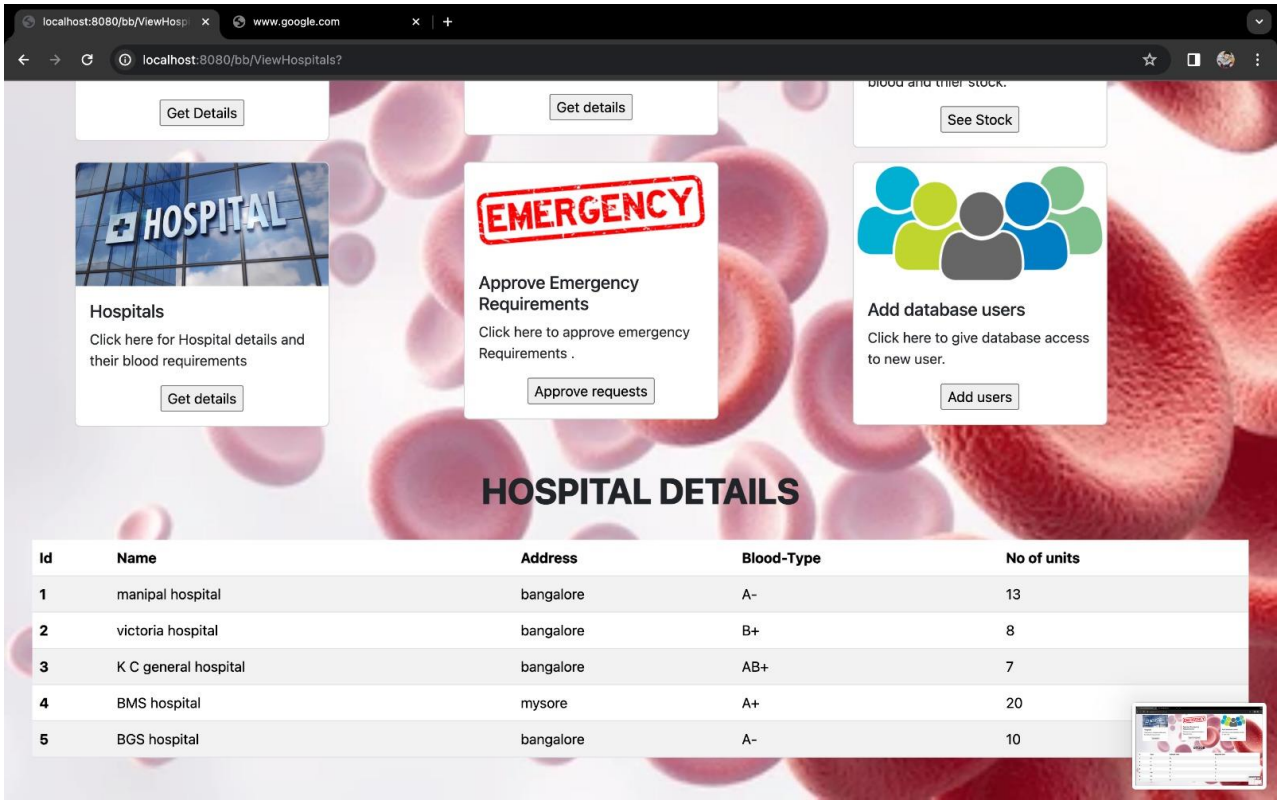


5. DONOR DETAILS

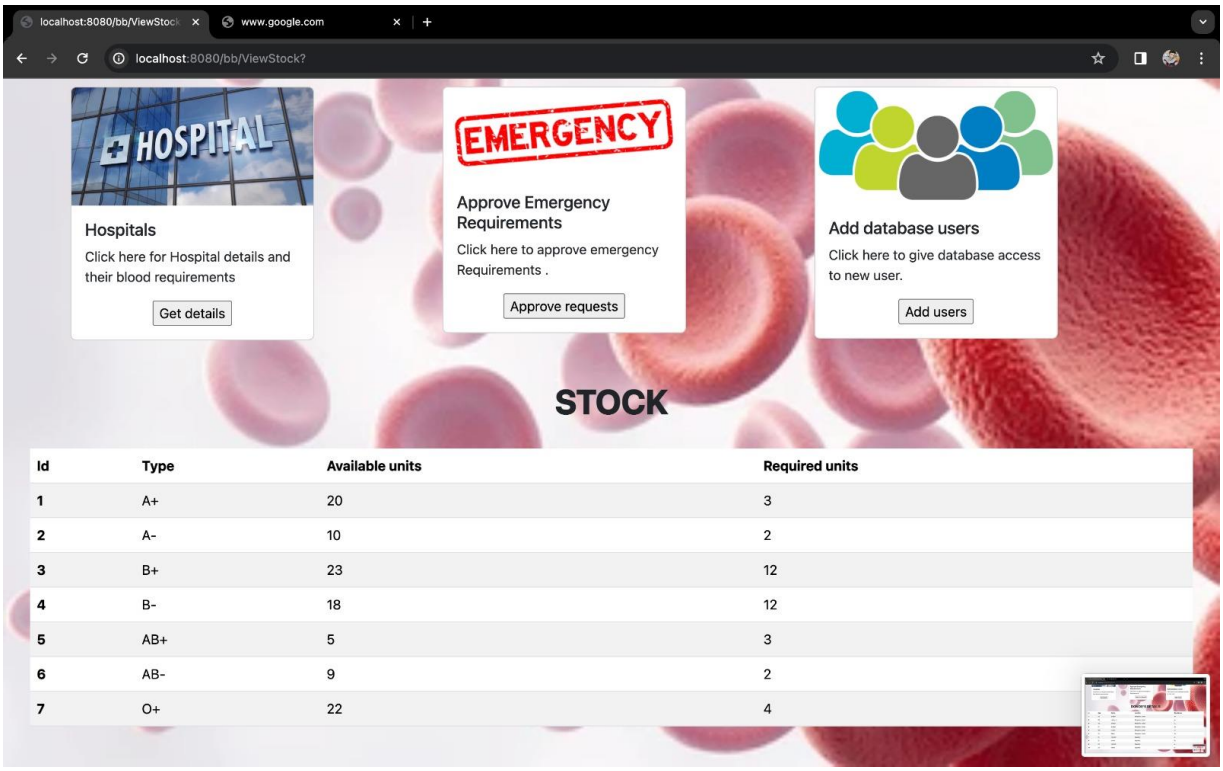


Id	Age	Name	Location	Bloodgroup
1	23	prajwal	Bengaluru Urban	A+
2	23	rahul p m	Bengaluru Urban	a+
3	23	prajwal	Bengaluru Urban	a_
4	21	prajwal	Bengaluru Urban	a+
5	100	mukthi	Bengaluru Urban	a+
6	22	Rahul	Bengaluru Urban	A+
7	21	harshith	Bagalkot	A-
8	21	shishir	Bagalkot	O+
9	22	harshith	Bagalkot	a-
10	23	lokesh	Bagalkot	a-

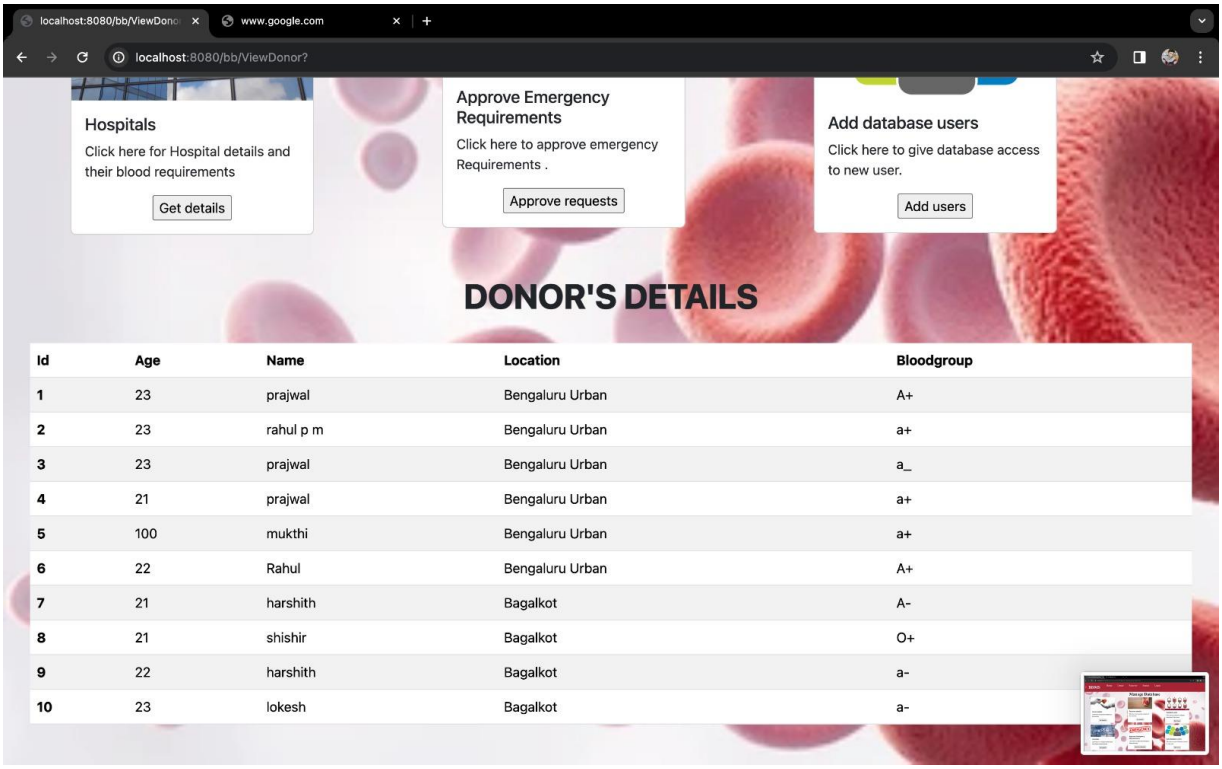
6. HOSPITAL DETAILS



7. BLOOD STOCK DETAILS

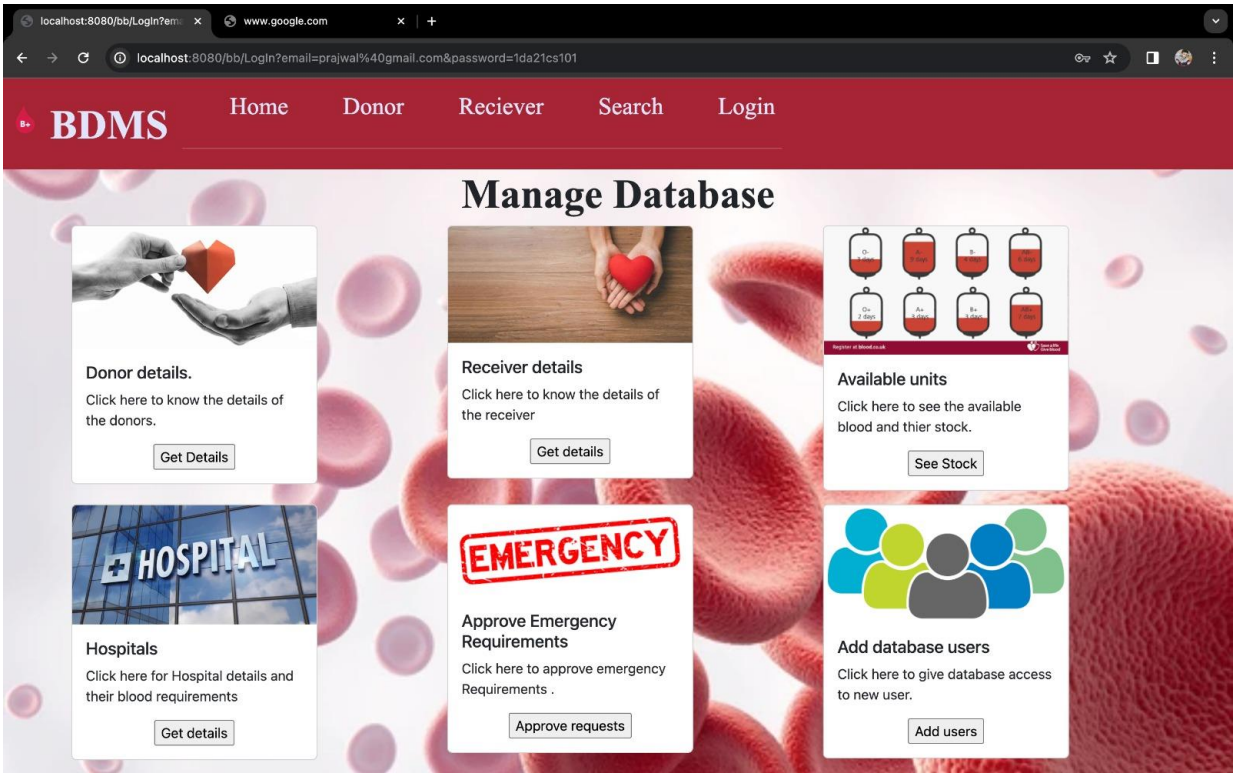


8. DONOR DETAILS

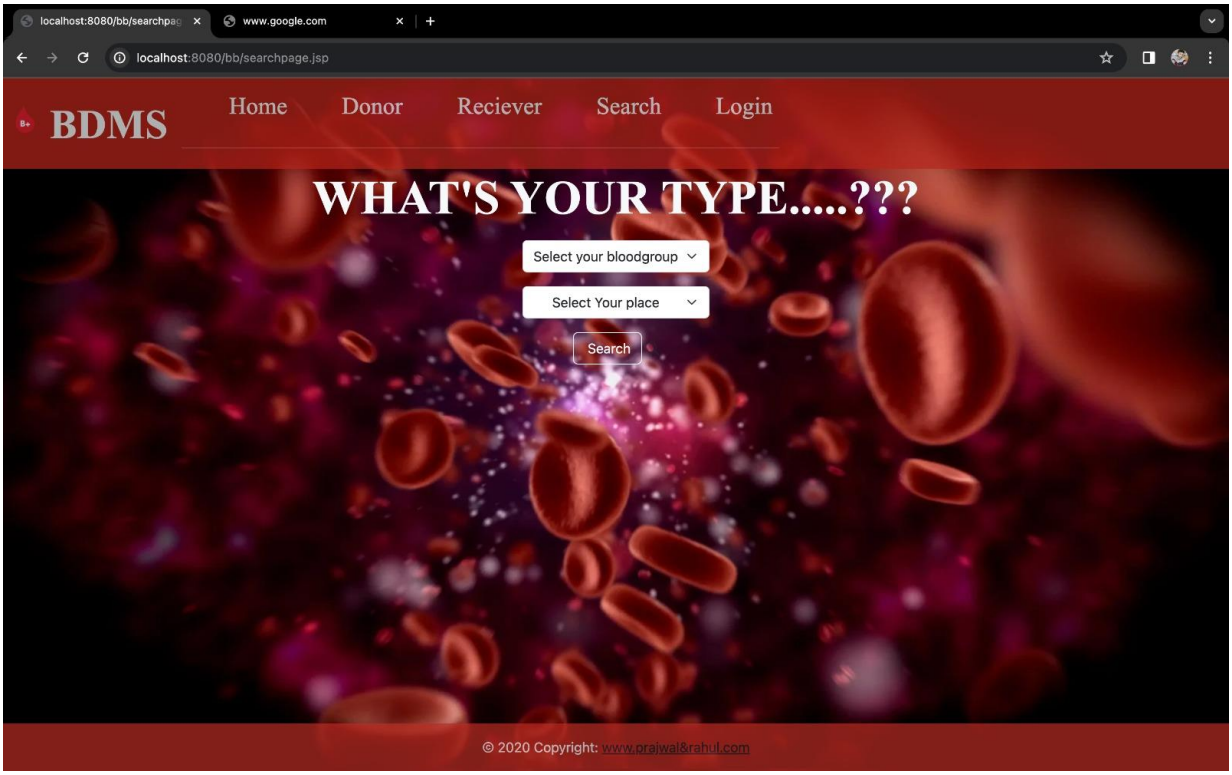


Id	Age	Name	Location	Bloodgroup
1	23	prajwal	Bengaluru Urban	A+
2	23	rahul p m	Bengaluru Urban	a+
3	23	prajwal	Bengaluru Urban	a_
4	21	prajwal	Bengaluru Urban	a+
5	100	mukthi	Bengaluru Urban	a+
6	22	Rahul	Bengaluru Urban	A+
7	21	harshith	Bagalkot	A-
8	21	shishir	Bagalkot	O+
9	22	harshith	Bagalkot	a-
10	23	lokesh	Bagalkot	a-

9. DASHBOARD



10. SEARCH BLOOD TYPE



CONCLUSION

A Blood Bank Database Management System is a robust solution designed to efficiently handle various aspects of blood bank operations, donor management, and administrative tasks. The system integrates multiple tables, including Admin, Donors, Blood Inventory, Donation Records, and Transfusion History, each serving distinct functionalities.

Admin functionalities encompass login, donor management, blood inventory monitoring, and the ability to view donation and transfusion records from the respective tables. Users engage in login, donor registration, blood donation appointments, and viewing their donation history, with seamless interaction between the Donors, Users, and Donation Records tables.

Key features include real-time updates to the Donation Records upon blood donation or transfusion and dependency on the Donors table for data integrity. The system ensures secure operations, requiring user authentication for donor registration and blood donation appointments. Admins manage blood inventory and donation records, facilitating efficient tracking and ensuring the availability of safe blood units. In conclusion, the Blood Bank Database Management System provides a comprehensive solution for efficient and secure blood bank operations, catering to both administrators and donors. The well-designed database structure and integrated functionalities contribute to streamlined processes, enhancing the overall efficiency and reliability of blood bank services.

BIBLIOGRAPHY

Books:

Ramez Elmasri and Shamkant B Navathe, "Fundamental Of Database System", Pearson Education Limited, Seventh Edition, 2015.

Websites:

https://dev.mysql.com/doc/

https://dev.mysql.com/doc/refman/8.0/en/

https://www.w3schools.com/mysql/

https://getbootstrap.com/docs/4.3/getting-started/introduction/

https://github.com/arkprocoder/

https://www.scribd.com/document/656265504/