**1. INTRODUCTION**

**1.1 OVERVIEW OF THE PROJECT**

The Title of Our Project is: ”Bood Hub”.

The Blood Hub is a Blood Bank Management System. It was chosen because it addresses a critical need in the healthcare industry. The current paper-based system is prone to errors and delays, which can have serious consequences for patients in need of blood transfusions. By automating the blood bank management process, we can improve the efficiency and accuracy of blood donation and inventory management, ultimately saving lives. Additionally, the project provides an opportunity to develop skills in web development, database management, and software engineering.

**1.2 OBJECTIVE OF THE PROJECT:**

* The main objective of the Blood Hub is to provide a web-based platform that streamlines the blood donation and inventory management process.
* The project aims to automate the process of blood donation, inventory management and requests.
* The system will help in saving lives by ensuring that blood is available when needed.
* Hospitals will be able to view the list of donors and request.
* Users will be able to view the list of hospitals and request.

**1.3 PROJECT CATEGORY**

Web-Based Application Using RDBMS

**1.4 TOOLS AND PLATFORM TO BE USED**

* Front-end: HTML, CSS, JavaScript
* Back-end: MySQL, PHP

**1.5 OVERVIEW OF THE TECHNOLOGIES USED**

The proposed system will undergo various testing techniques such as unit testing, systems integration test, and user-interface tests. The testing plan will be created based on the requirements and design documents

**1.5.1 Hardware Requirements**

* Processors: Intel Pentium dual-core or above
* RAM: 2GB and above
* Hard disk Utilization: 40GB and above
* Input Devices: Mouse, Keyboard

**1.5.2 Software Requirements**

* Browser: Internet Explorer, Google Chrome, Mozilla Firefox
* Server: Apache
* IDE: Sublime Text Editor

**1.5.3 Front-End**

Front-end development refers to the process of creating the user interface and user experience of a website or web application. It involves designing and building the visual elements that users interact with directly in their web browsers. The three core technologies used in front-end development are HTML, CSS, and JavaScript

**1.5.4 Back-End**

Back-end development is the part of web development that deals with the server-side logic and database management of a web application. It focuses on handling requests from the front-end, processing data, and delivering responses. In this context, let's explain MySQL and PHP, two commonly used technologies for back-end development

**1.6 ORGANIZATION PROFILE**

Organization Name: XYZ Solutions

Mission Statement:

At XYZ Solutions, we are dedicated to improving the accessibility, efficiency, and effectiveness of blood donation and distribution systems. Our mission is to save lives by providing a robust and user-friendly Blood Bank Management System that streamlines the process of blood collection, testing, storage, and distribution.

About Us:

XYZ Solutions is a leading software development company specializing in healthcare and medical solutions. With a team of experienced developers, healthcare experts, and project managers, we are committed to delivering innovative and reliable software solutions to address critical healthcare challenges.

At XYZ Solutions, we believe that our Blood Bank Management System, Blood Hub will make a significant impact in the healthcare sector by improving the management of blood resources and ultimately saving lives. We are committed to partnering with blood banks and healthcare institutions to achieve this noble goal.

**1.7 STRUCTURE OF THE PROGRAM**

The Blood Hub consists of two modules: Hospital Module, and User Module

**HOSPITAL :**

* Login : Hospital has to login using his credentials.
* Manage Blood Stock: Hospital can add, delete and view blood stocks.
* Manage Requests : Hospital can approve or reject recipient’s requests.
* Request Blood : Hospital can view and request to registered donors.
* Blood Status : Hospital can view their status of blood requests.
* Manage Profile : The hospital can add, edit and view their profile.

**USERS :**

* Register : The user can register by giving their basic details.
* Login : The user can login using username and password.
* Add Blood: The user can add and view the blood list.
* Request Blood : The user can view the available blood list in hospitals and request.
* Status of Request : The user can check their status of request.
* Manage Request: The user can approve or reject the blood donation request.
* Manage Profile : The user can add, edit and view their profile.

**1.8 STATEMENT OF THE PROBLEM**

Blood banks play a vital role in saving lives, but managing blood donations and inventory can be a complex and challenging task. The current paper-based system is inefficient and time-consuming, and there is a need for a more streamlined and automated approach to blood bank management. The Blood Bank Management System aims to solve this problem by providing a web-based platform for managing blood donations, inventory, and requests.

**2. SOFTWARE REQUIREMENTS SPECIFICATION**

**2.1 INTRODUCTION**

This Software Requirement Specification document provides a complete description of all the functionalities and the specifications of the “Blood Hub” system. The following section provides an overview of the derived Software Requirements Specification (SRS) for the application. To begin with, the purpose of the document is presented, and its intended audience is outlined. Subsequently, the scope of the project specified by the document is given with a particular focus on what the resultant software will do and the relevant benefits associated with it. The nomenclature used throughout the SRS is also offered. To conclude, a complete document overview is provided to facilitate increased reader comprehension and navigation.

**2.1.1 Purpose**

The Blood Hub is a Blood Bank Management System. It was chosen because it addresses a critical need in the healthcare industry. The current paper-based system is prone to errors and delays, which can have serious consequences for patients in need of blood transfusions. By automating the blood bank management process, we can improve the efficiency and accuracy of blood donation and inventory management, ultimately saving lives. Additionally, the project provides an opportunity to develop skills in web development, database management, and software engineering

**2.1.2 Scope of the project**

The scope of the project is to develop the Hospital and User modules to manage donors and manage blood requests.

* The project aims to automate the process of blood donation, inventory management and requests.
* The system will help in saving lives by ensuring that blood is available when needed.
* Hospitals will be able to view the list of donors and request.
* Users will be able to view the list of hospitals and request.

**2.1.3 Intended Audience and Reading Suggestions**

The intended audience for a discussion on blood management can vary widely, encompassing healthcare professionals, administrators, policymakers, and patients alike. Healthcare providers, including doctors, nurses, and laboratory technicians, need to stay updated on the latest techniques and guidelines for optimizing blood transfusions, minimizing blood loss during surgeries, and effectively managing anemia. Hospital administrators and policymakers play a crucial role in shaping blood management policies and ensuring efficient resource allocation. Additionally, patients and their families benefit from understanding the importance of blood conservation and being informed about their options for managing blood-related conditions. To cater to this diverse audience, it's essential to provide a range of reading materials.

**2.1.4 Definitions, Acronyms and Abbreviations**

* GUI - Graphical User Interface
* DBMS - Database Management System
* RDBMS - Relational Database Management System
* SRS - Software Requirement Specification
* CPU -Central processing unit
* PHP -Hypertext Preprocessor.
* SQL -Structured Query Language.
* HTML -Hyper Text Markup Language.
* CSS -Cascading style sheet Top of Form

**2.2 OVERALL DESCRIPTION**

This section will give an overview of the whole system. The system will be explained in its context to show how the system interacts with other systems and introduce its basic functionality of it. It will also describe what type of users will use the system and what functionality is available for each type. At last, the constraints and assumptions for the system will be presented.

**2.2.1 Product Perspective**

Blood Hub is an innovative approach to managing blood donation and inventory. The system's main achievement is providing a user-friendly interface that allows hospitals and donors to easily manage their accounts and inventory. Overall, the proposed system will streamline the blood donation process and help save lives.

**2.2.2 Product Features**

In the future this project can be implemented through kiosk model.

**2.2.3 User Characteristics**

The system consists of two users:

**HOSPITAL:**

The Hospital can log in to the system using his/her unique credential. Here Hospital can manage their profile, blood stock, requests and can request for blood and can keep track of their request.

* Login
* Manage Blood Stock
* Manage Requests
* Request Blood
* Blood Status
* Manage Profile

**USERS:**

The Users can log in by entering their username and password. The Users can view the available blood stock and can request, manage requests and their profiles, and also can track their status of request.

* Register
* Login
* Add Blood
* Request Blood
* Status of Request
* Manage Request
* Manage Profile

**2.3 OPERATING ENVIRONMENT**

**2.3.1 Design and Implementation Constraints**

* While user or hospital register to the system, mandatory fields must be checked for validation whether the user or hospital has filled appropriate data in these mandatory fields. If not, a proper error message should be displayed, or else the data is to be stored in a database for later retrieval.
* All mandatory fields should be filled by the hospitals/users while adding the blood stocks.
* All mandatory fields should be filled by the hospitals/users while requesting the blood.
* The system must be designed in such a way that will be easy to use and visible on most browsers.

**2.3.2 General Constraints**

General constraints include the following:

* This application requires an internet connection
  + - * Only hospital can manage all the requests.
      * No one has the right to change the informations in the website.
      * The end system should also allow for seamless recovery, without data loss, from individual device failure.

**2.3.3 Assumptions and Dependencies**

* It is assumed that system has 2 types of Users, i.e., Users and Hospitals.
* The user should be careful when registering the details.
* All the details related to user and hospital should be maintained properly.
* All the data entered will be correct and up to date.
* It is assumed that the needed changes, to collect and store the data, will be made within the current application and database.

**2.4 SPECIFIC REQUIREMENTS**

**2.4.1 External Interface Requirements**

**2.4.1.1User Interface**

A user interface is a point of human interaction and communication with the system.

We have taken the following requirements during design,

• Textboxes to enter details.

* + - * + Buttons to add, delete, update and search.

• Labels to display the information.

• Checkboxes.

• Combo boxes and list boxes.

• Grid box to display the information.

**2.4.1.2 Hardware, Software and Communication Interface**

**Hardware Interface**

* Processors: Intel Pentium dual-core or above
* RAM: 2GB and above
* Hard disk Utilization: 40GB and above
* Input Devices: Mouse, Keyboard

**Software Interface**

* Browser: Internet Explorer, Google Chrome, Mozilla Firefox
* Server: Apache
* IDE: Sublime Text Editor

**Communication Interface**

The communications function required by this product is HTTP protocol, and internet communication is through TCP/IP protocol.

**2.4.2 Functional Requirements**

**Hospital Module:**

* **Login:** Hospital has to login using his credentials.
* **Manage Blood Stock:** Hospital can add, delete and view blood stocks.
* **Manage Requests**: Hospital can approve or reject recipient’s requests.
* **Request Blood**: Hospital can view and request to registered donors.
* **Blood Status**: Hospital can view their status of blood requests.
* **Manage Profile**: The hospital can add, edit and view their profile.

**User Module**

* **Register:** The user can register by giving their basic details.
* **Login:** The user can login using username and password.
* **Add Blood**: The user can add and view the blood list.
* **Request Blood**: The user can view the available blood list in hospitals and request.
* **Status of Request**: The user can check their status of request.
* **Manage Request**: The user can approve or reject the blood donation request.
* **Manage Profile**: The user can add, edit and view their profile.

**2.4.3 Performance Requirements**

* The server shall be capable of supporting an arbitrary number of active owner payment, that is, no payments shall be lost under any circumstances.
* Page load time should be less than 40 sec.
* Should have a good memory space.
* The server shall be capable of supporting an arbitrary number of active orders, that is, no orders shall be lost under any circumstances.

Should be error-free.

* 1MB file should get uploaded in 60 sec.

**2.4.4 Design Constraints**

While user or hospital register to the system, mandatory fields must be checked for validation whether the user or hospital has filled appropriate data in these mandatory fields. If not, a proper error message should be displayed, or else the data is to be stored in a database for later retrieval.

• All mandatory fields should be filled by the owner while paying the fine details.

• All mandatory fields should be filled by the insurance company while adding the insurance details.

• The system must be designed in such a way that will be easy to use and visible on most browsers.

**2.4.5 Other Requirements**

**Safety Requirements:**

* There are two user levels in “BLOOD HUB”. Access to the various subsystems will be protected by a user log-in screen that requires a username and password. This gives different views and accessible functions of user levels through the system.
* Email ID once registered to the system cannot be changed to make every user unique and easily identifiable
* Maintaining backups ensures the system database security. The system can be restored in any case of an emergency.

**Security Requirements:**

* The server on which the ‘blood hub’ resides will have its security to prevent unauthorized write/delete access. There is no restriction on reading access.
* The proposed website will be secure. There are different categories of users they are users and hospitals.
* Depending upon the category of using the access rights are decided.

**3. SYSTEM ANALYSIS AND DESIGN**

**3.1 INTRODUCTION**

The purpose of the design process is to create a model or representation of a system that can be used to construct the system afterwards. The process of establishing the architecture, modules, interfaces, and data for a system in order to meet certain criteria is known as systems design. The goal of system design is to determine which modules are required for the system, their requirements, and how they should be linked.

The software analysis and system design mainly deal with the software development activity. The website “BLOOD HUB” design document explains how exactly website will work. It includes Context Flow Diagram, Data Flow Diagrams, ER Diagram. Context Flow Diagram explain the interaction between application and user and functionalities performed. The architecture explains overall functionality of the system and the sequence diagram explains the data flow between the entities.

**3.2 DATA FLOW DIAGRAM**

Data Flow Diagram is a graphical representation of a system or a portion of the system. It consists of data flows, process, sources and sink and stores all the description through the use of easily understandable symbols.

DFD is one of the most important modelling tools. It is used to model the system, components that interact with the system, uses the data and information flows in the system.

DFD shows the information moves through the and how it is modified by a series of transformations. It is a graphical technique that depicts information moves from input or output.

DFD is also knows as bubble chart or Data Flow Graphs. DFD may be used to represent the system at any level of abstraction. DFD’s may partition into a level that represents increasing information flows and functional details.

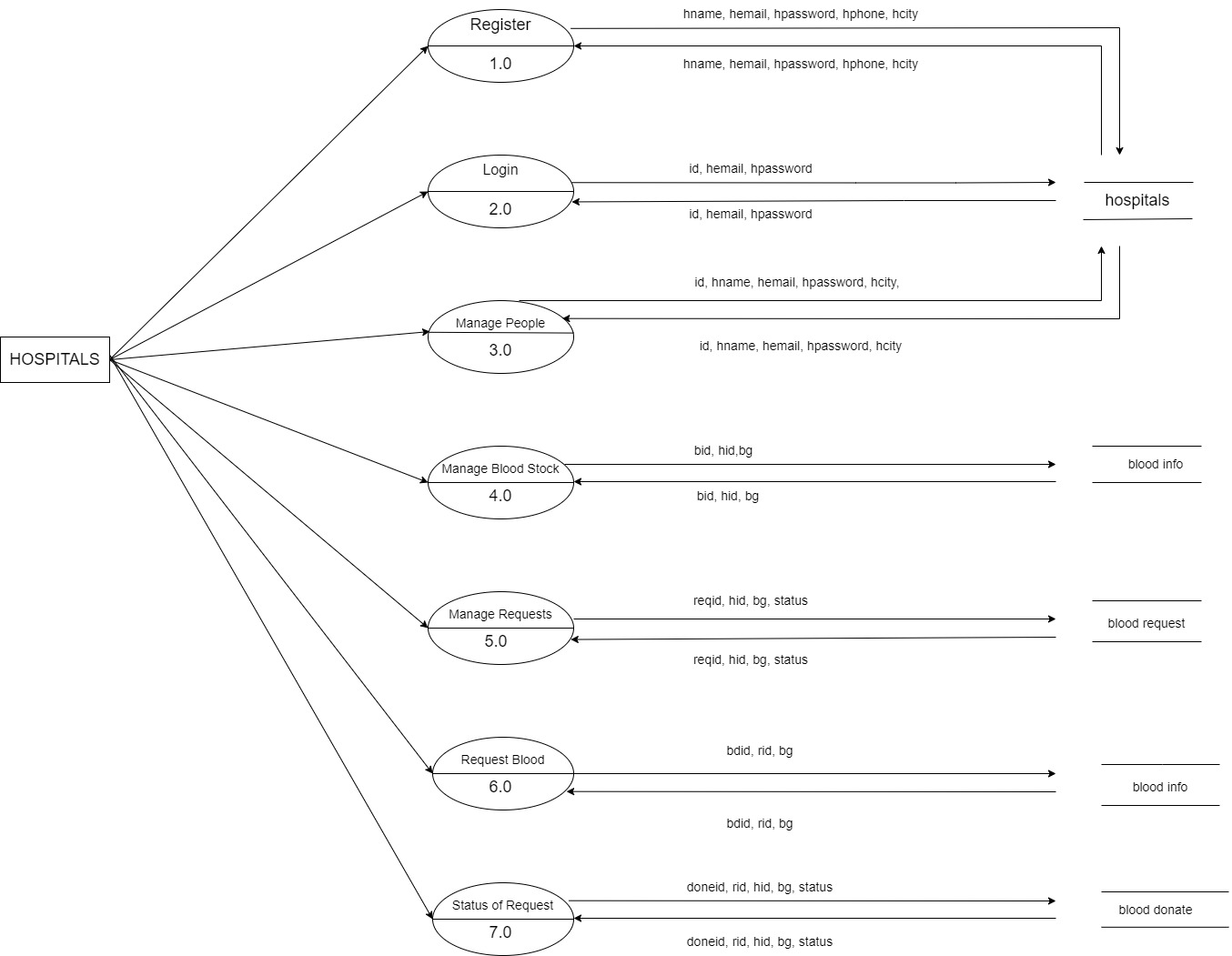
Rules Regarding DFD Construction:

* A process cannot have only outputs.
* A process cannot have only inputs.
* The inputs to a process must be sufficient to produce the outputs from the process. • All data stores must be connected to at least one process.
* All data stores must be connected to a source or sink.
* A data flow can have only one direction of flow. Multiple data flows to and/or from the same process and data store must be shown by separate arrows.
* If the exact same data flows to two separate arrows, it should be represented by a forked arrow.
* Data cannot flow directly back into the process it has just left. All data flows must be named using a noun phrase.

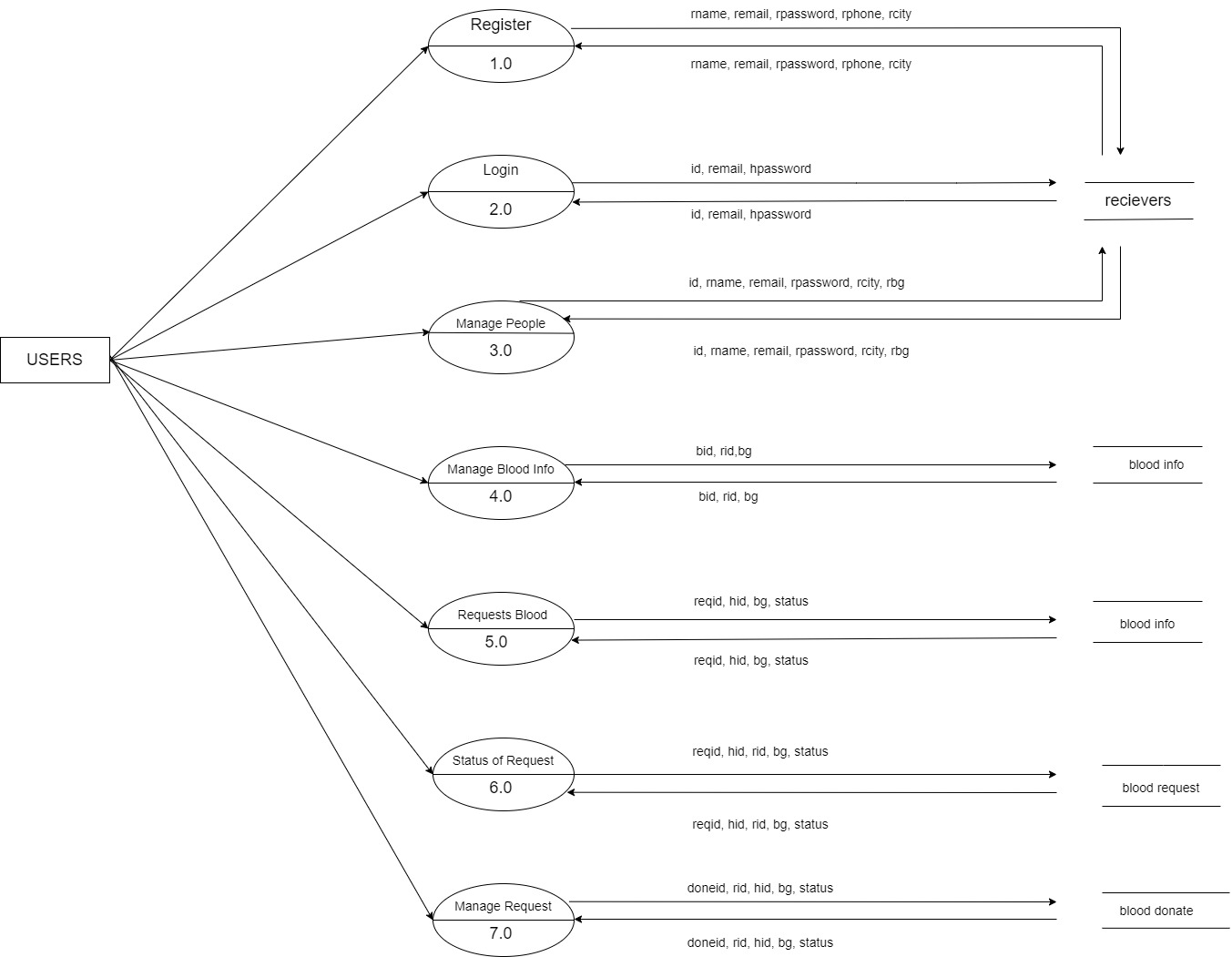
**DFD Symbols:**

|  |  |  |
| --- | --- | --- |
| **Name** | **Notation** | **Description** |
| **Process** |  | A process transforms incoming data flow into outgoing data flow. The processes are shown by named circles. |
| **Datastore** |  | Data stores are repositories of data in the system. They are sometimes also referred to as files. |
| **Dataflows** |  | Data flows are pipelines through which packets of information flow. Label the arrows with the name of the data that moves through it. |
| **External Entity** |  | External entities are objects outside the system with which the system communicates. External Entities are sources and destinations of the system’s inputs and outputs |

**DFD level 1(Hospital)**

****

**DFD level 1(User)**

****

**3.3 DATA BASE DESIGN**

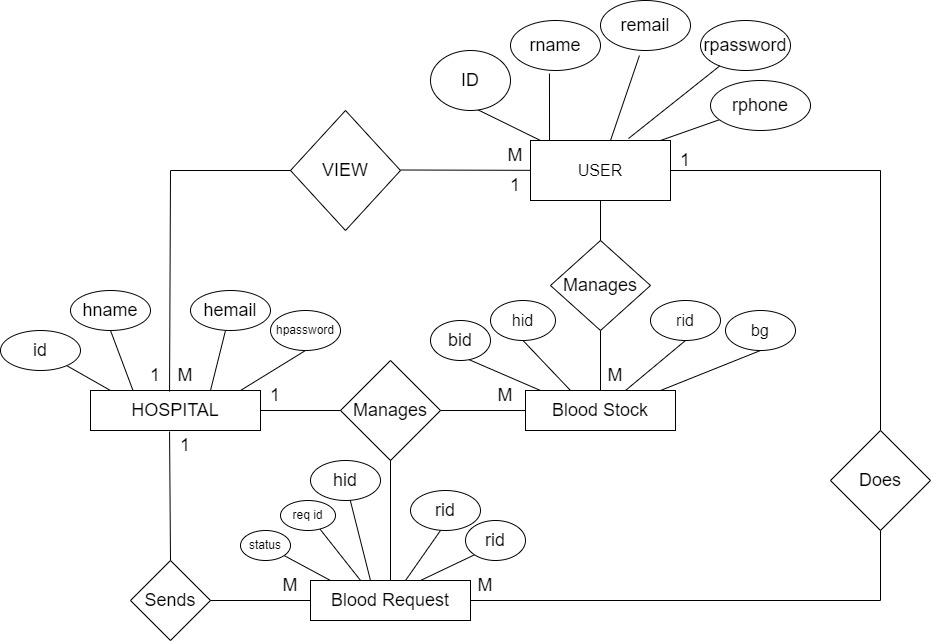
**3.3.1 Entity-Relationship Diagram**

**ER Diagram** stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.

**Entity**: An entity is an object or component of data. An entity is represented as rectangle in an ER diagram.

**Attribute:** An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

* Key attribute
* Composite attribute
* Multivalued attribute
* Derived attribute
* **Key attribute:** A key attribute can uniquely identify an entity from an entity set. Key attribute is represented by oval same as other attributes however the text of key attribute is underlined.
* **Composite attribute:** An attribute that is a combination of other attributes is known as composite attribute.
* **Multivalued attribute:** An attribute that can hold multiple values is known as multivalued attribute. It is represented with double ovals in an ER Diagram.
* **Derived attribute**: A derived attribute is one whose value is dynamic and derived from another attribute. It is represented by dashed oval in an ER Diagram.
* **Relationship:** A relationship is represented by diamond shape in ER diagram, it shows the relationship among entities. There are four types of relationships:
* One to One
* One to Many
* Many to One
* Many to Many
* **One to One Relationship:** When a single instance of an entity is associated with single instance of another entity then it is called one to one relationship.
* **One to Many Relationship:** When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship.
* **Many to One Relationship:** When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship.
* **Many to Many Relationship:** When more than one instances of an entity is associated with more than one instances of another entity then it is called many too many relationship.



**3.3.2** **Table Relationship:**

**3.3.3 Table Description:**

A table is an arrangement of data in rows and columns, or possibly in a more complex structure. The database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyse the data. A database consists of one or more tables. Each table is made up of rows and columns. Each row in a relational table is uniquely identified by a primary key. This can be by one or more sets of column values. In most scenarios it is a single column, such as student ID. Every relational table has one primary key. Its purpose is to uniquely identify each row in the database. No two rows can have the same primary key value. The practical result of this is that you can select every single row by just knowing its primary key.

**Table Name: Hospital**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL.NO** | **Attribute Name** | **Data Type(Size)** | **Constraints** | **Description** |
| 1 | id | int(11) | Primary Key | Hospital ID |
| 2 | hname | varchar(100) | Not Null | Hospital Name |
| 3 | hemail | varchar(100) | Not Null | Hospital Email |
| 4 | hpassword | Varchar(100) | Not Null | Hospital Password |
| 5 | hphone | Varchar(100) | Not Null | Hospital phone |
| 6 | hcity | Varchar(100) | Not Null | Hospital city |

**Table Name : receivers**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL.NO** | **Attribute Name** | **Data Type(Size)** | **Constraints** | **Description** |
| 1 | id | int(11) | Primary Key | Receiver ID |
| 2 | rname | Varchar(100) | Not Null | Receiver name |
| 3 | remail | Varchar(100) | Not Null | Receiver email |
| 4 | rpassword | Varchar(100) | Not Null | Receiver password |
| 5 | rphone | Varchar(100) | Not Null | Receiver phone |
| 6 | rbg | Varchar(10) | Not Null | Receiver blood group |
| 7 | rcity | Varchar(100) | Not Null | Receiver city |

**Table Name: blooddinfo**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL.NO** | **Attribute Name** | **Data Type(Size)** | **Constraints** | **Description** |
| 1 | bdid | int(11) | Primary Key |  |
| 2 | rid | Int(11) | Foreign Key | Receiver ID |
| 3 | bg | varchar(10) | Not Null | Blood group |
| 4 | oname | varchar(30) | Not Null | Rest Owner Name |
| 5 | email | varchar(30) | Not Null | Email ID |
| 6 | mob | int(10) | Not Null | Mobile no |
| 7 | pass | Varchar(10) | Not Null | Password |

**Table Name : blooddonate**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL.NO** | **Attribute Name** | **Data Type(Size)** | **Constraints** | **Description** |
| 1 | donoid | int(11) | Primary Key | Donor ID |
| 2 | rid | Int(11) | Foreign Key | Receiver ID |
| 3 | hid | Int(11) | Not Null | Hospital ID |
| 4 | bg | varchar(11) | Not Null | Blood group |
| 5 | status | Varchar(100) | Not Null | status |
| 6 | password | Int(10) | Not Null | Password |

**Table Name : bloodinfo**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL.NO** | **Attribute Name** | **Data Type(Size)** | **Constraints** | **Description** |
| 1 | bid | int(11) | Primary Key | Blood ID |
| 2 | hid | int(11) | Foreign Key | Hospital ID |
| 3 | bg | varchar(10) | Not Null | Blood group |

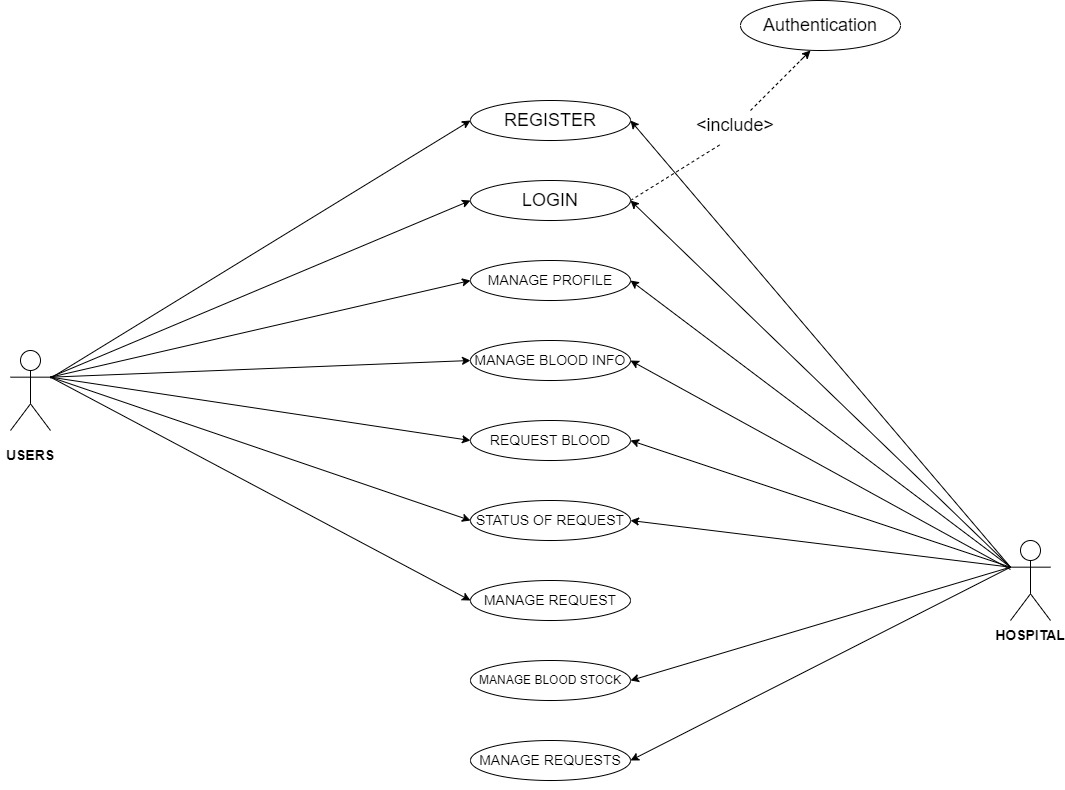
**Table Name : bloodrequest**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SL.NO** | **Attribute Name** | **Data Type(Size)** | **Constraints** | **Description** |
| 1 | reqid | int(11) | Primary Key | Request ID |
| 2 | hid | Int(11) | Foreign Key | Hospital ID |
| 3 | rid | Int(11) | Not Null | Receiver ID |
| 4 | bg | varchar(11) | Not Null | Blood group |
| 5 | status | varchar(100) | Not Null | status |

**3.4 System Design Implementation**

**3.4.1 Use Case:**

A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real-world objects and systems. There are a number of benefits with having a use case diagram over similar diagrams such as flowcharts.



**3.4.2 Class Diagram:**

**Introduction**

A Class diagram in the Unified Modeling Language is a type of static structure that describes the structure of a system by showing the system’s classes, their attributes, operations and the relationships among the objects. The main purpose of class diagrams is to build a static view of an application. It is the only diagram that is widely used for construction, and it can be mapped with object-oriented languages. A class diagram issued to visualize, describe, document various different aspects of the system, and also construct executable software code.

**A class is represented by a rectangle having three sections –**

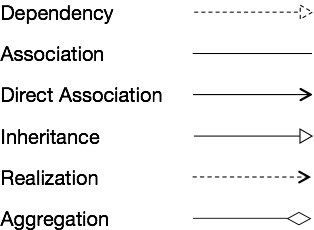
* The top section containing the name of the class
* The middle section containing class attributes
* The bottom section representing operations of the class

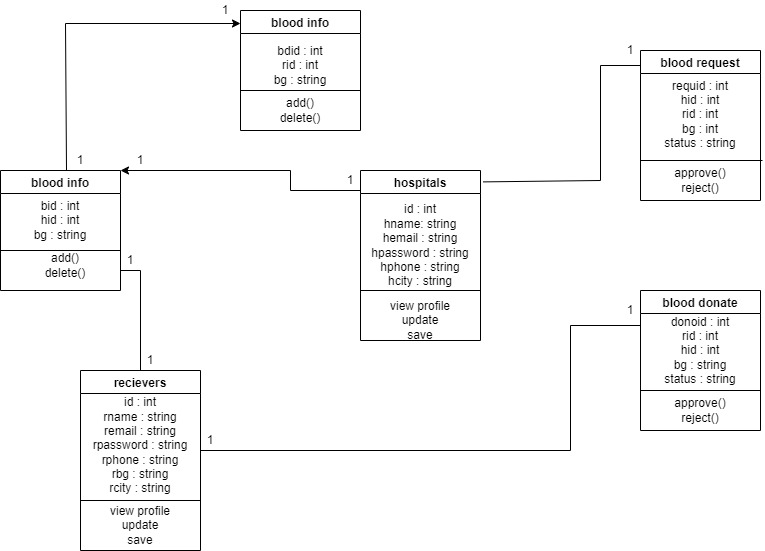
**Visibility**

The visibility of the attributes and operations can be represented in the following ways −

* **Public** − A public member is visible from anywhere in the system. In class diagram, it is prefixed by the symbol ‘+’.
* **Private** − A private member is visible only from within the class. It cannot be accessed from outside the class. A private member is prefixed by the symbol ‘−’.
* **Protected** − A protected member is visible from within the class and from the subclasses inherited from this class, but not from outside. It is prefixed by the symbol ‘#’.
* **Notation**

The notations for the different types of relationships are as follows –

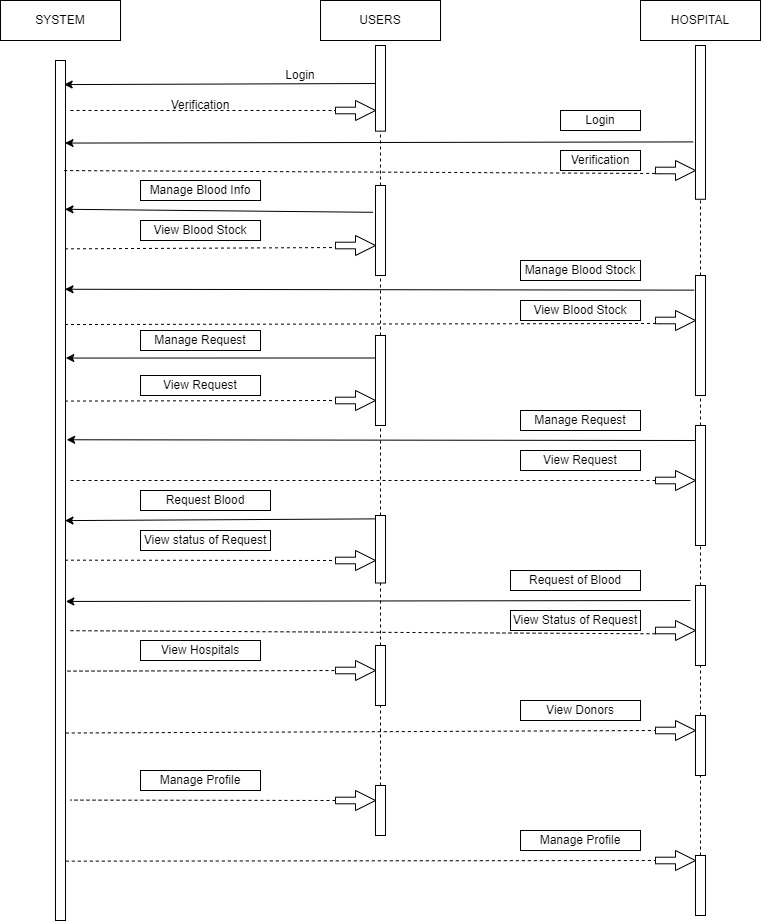




**3.4.3 Sequence Diagram:**

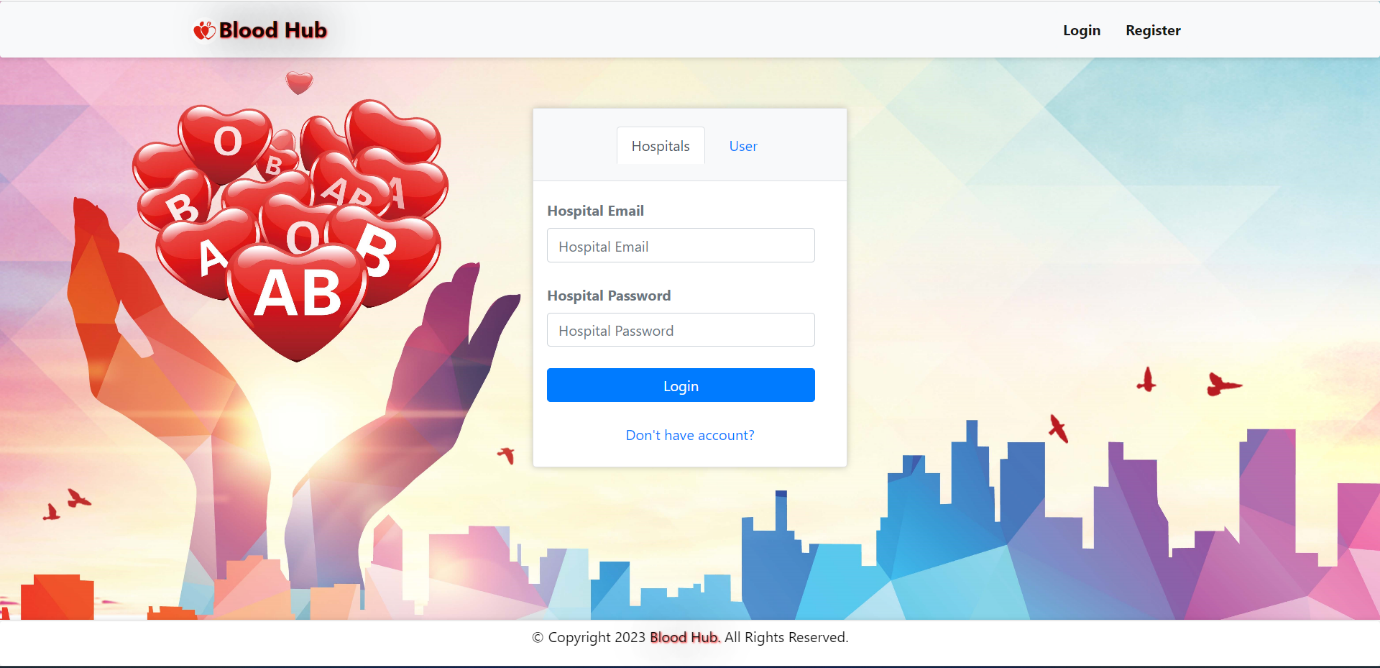
Sequence diagram model is the flow of logic within the system in a visual manner, enabling both to document and validate the logic and are commonly used for both analysis and design purposes. It is the most popular UML artifact for dynamic modeling which focus on identifying the behavior within the system. Sequence diagram are typically associated with use case realizations in the logical view of the system under development. Sequence diagrams are sometime called as event diagrams and event scenarios.

A sequence diagram shows, as parallel vertical lines different processes or the objects that live simultaneously and as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. A sequence diagram shows object interactions arranged in time sequence, it depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.



**3.5 User Interface Design:**

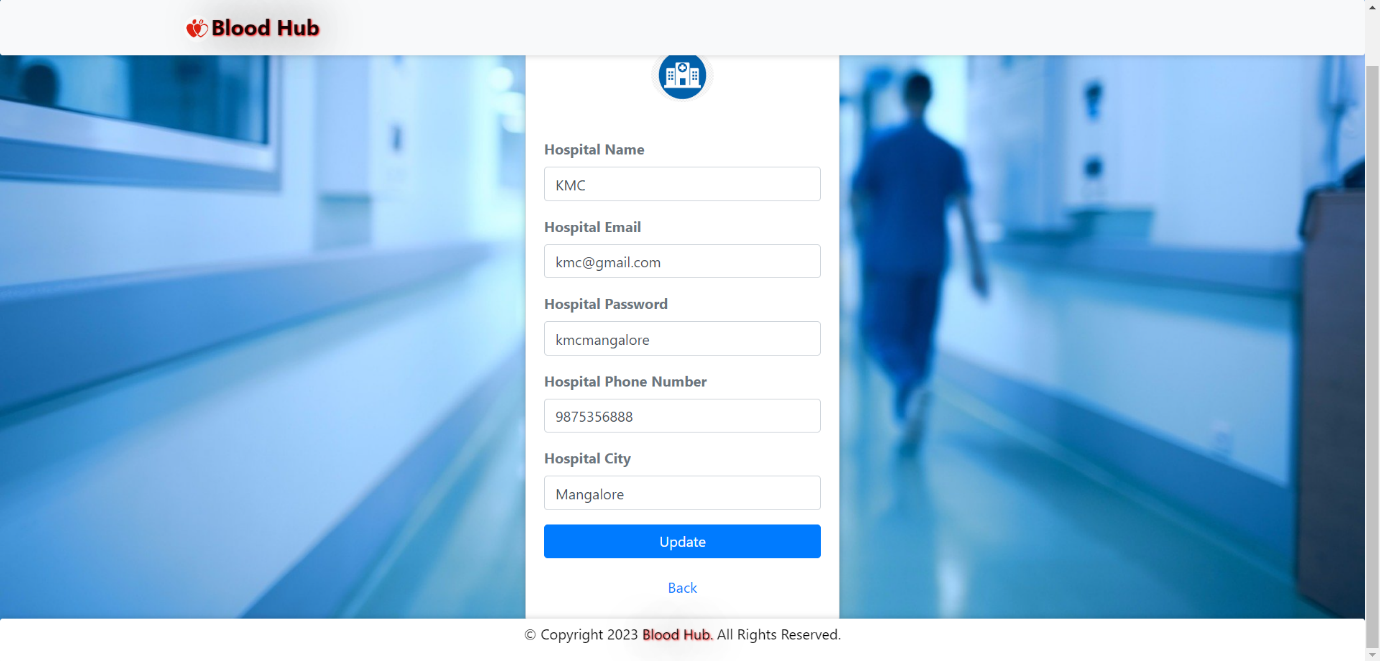
**HOSPITAL LOGIN**

****

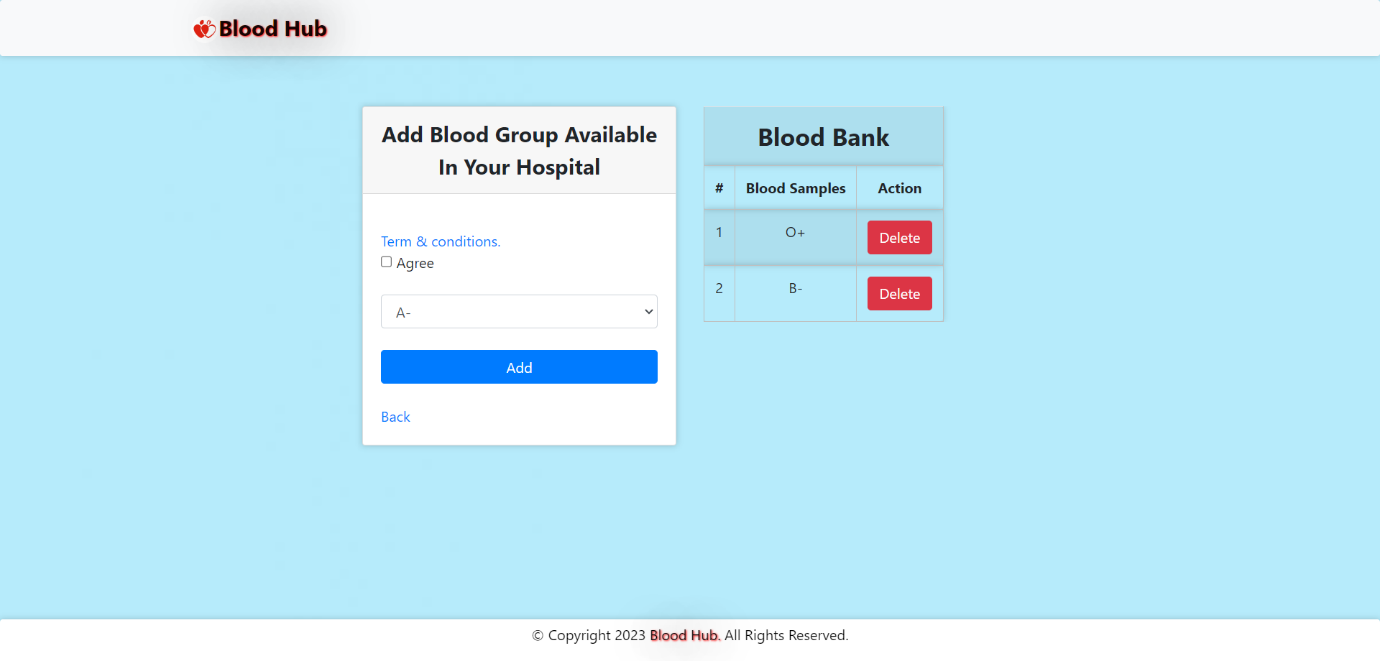
**HOSPITAL DASHBOARD**

****

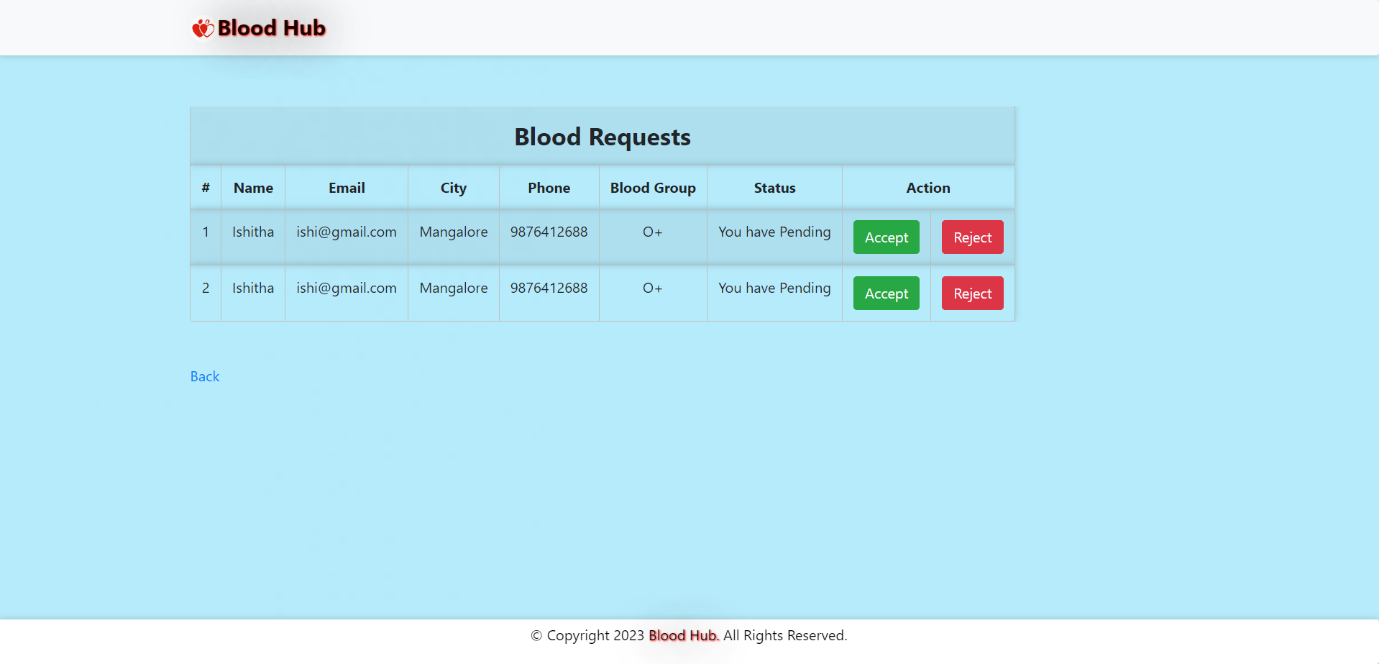
**MANAGE PROFILE**

****

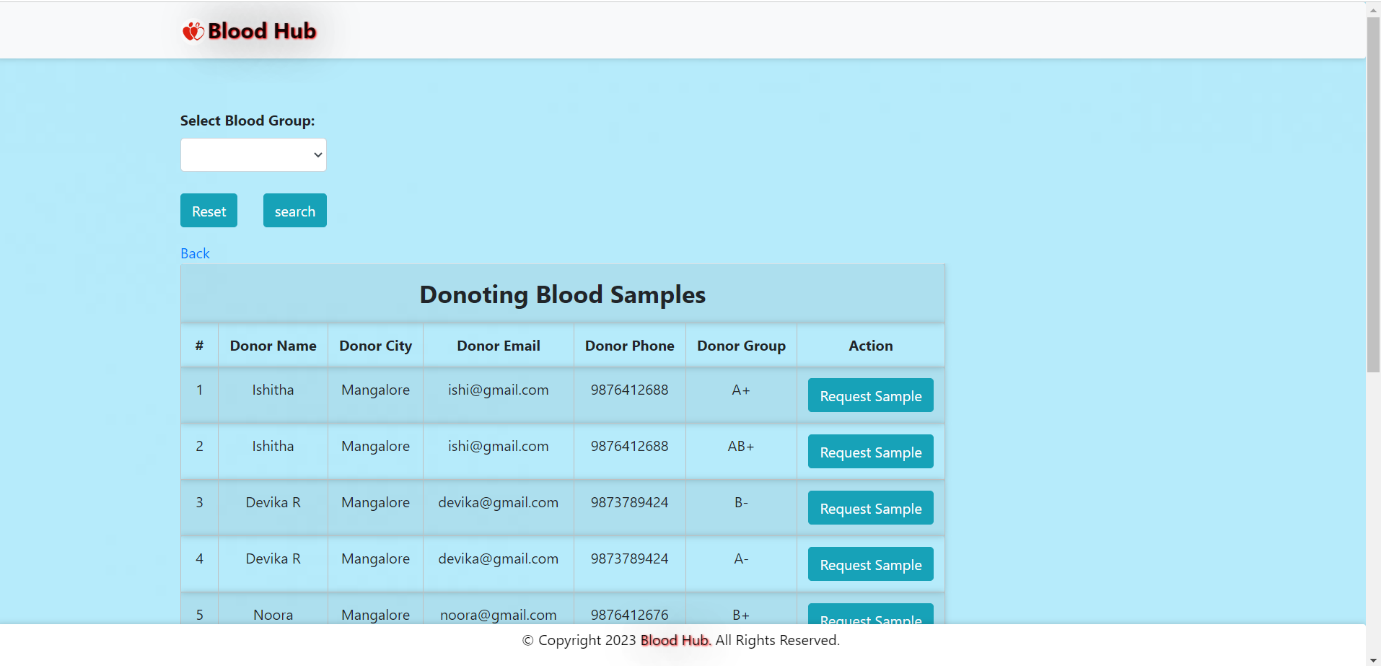
**ADD BLOOD**

****

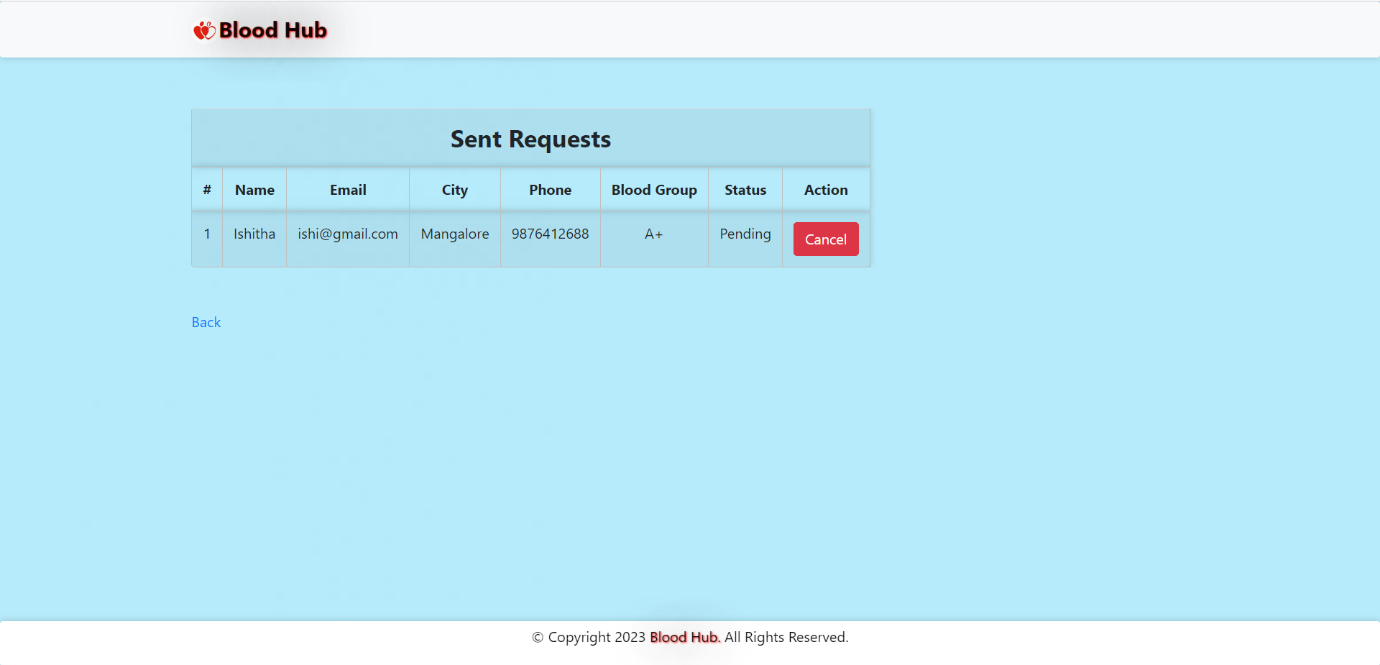
**BLOOD REQUESTS**

****

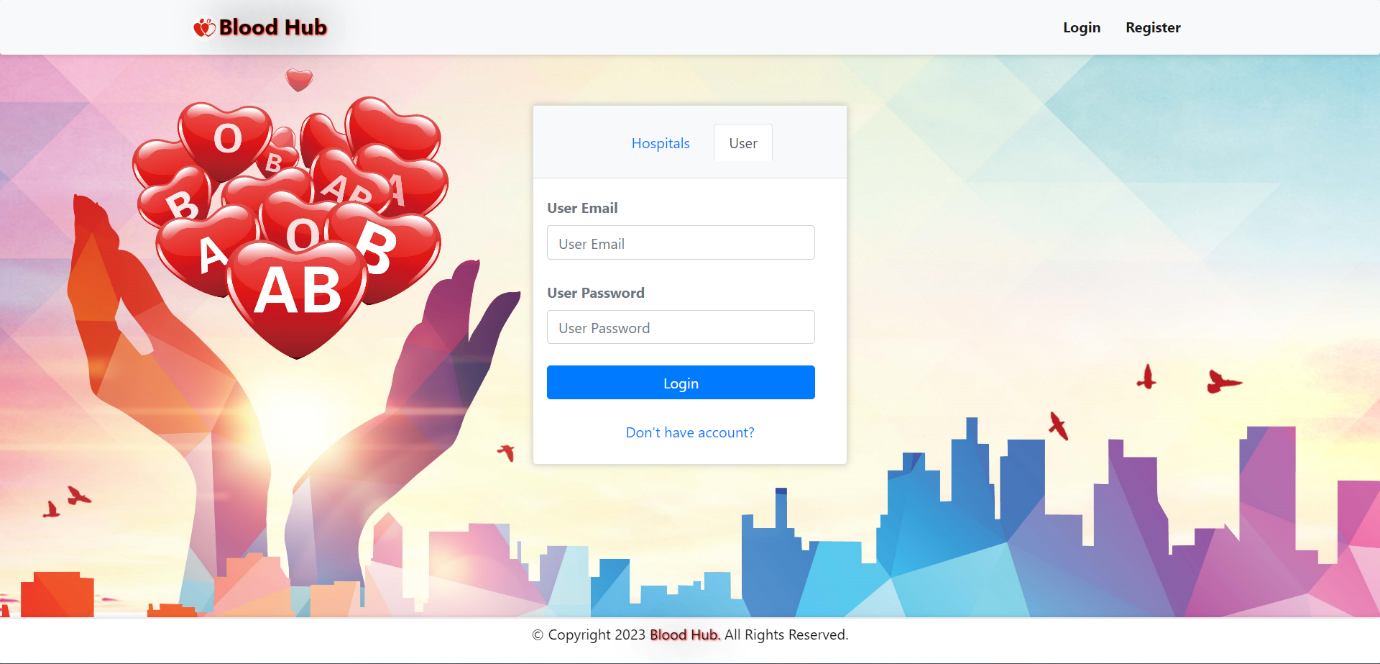
**NEED BLOOD**

****

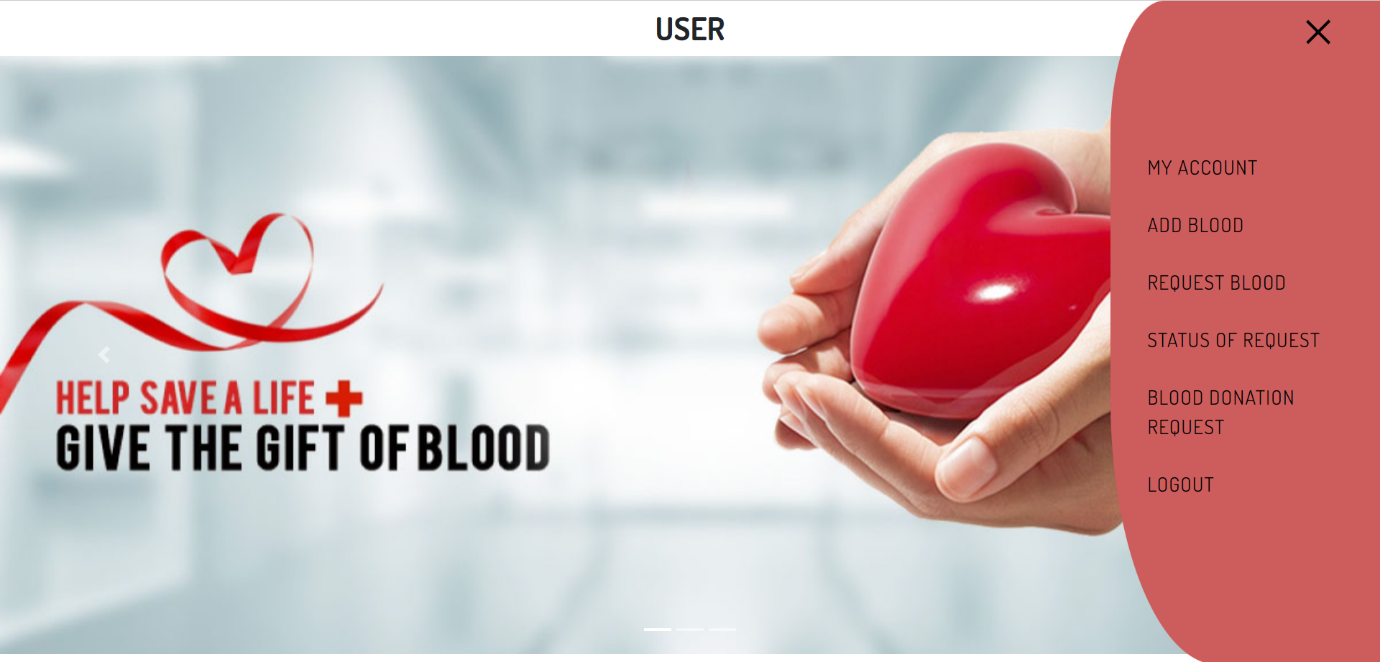
**STATUS OF YOUR BLOOD REQUEST**

****

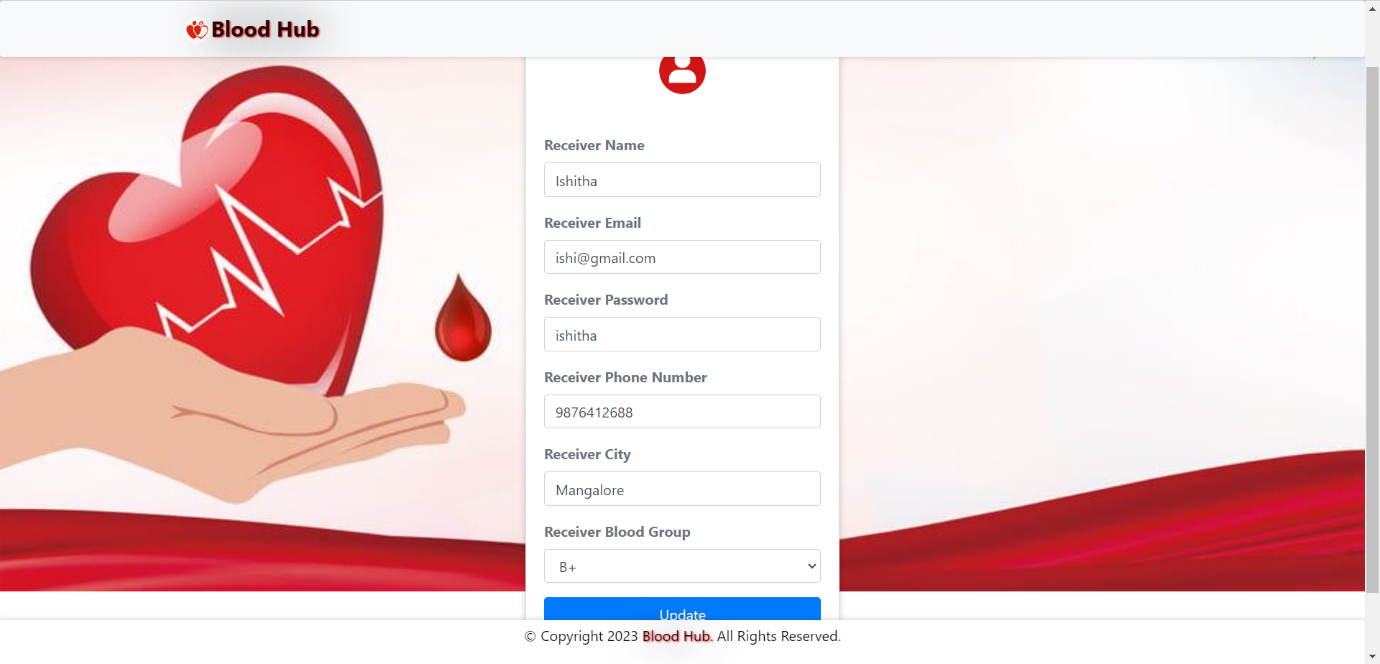
**USER LOGIN**

****

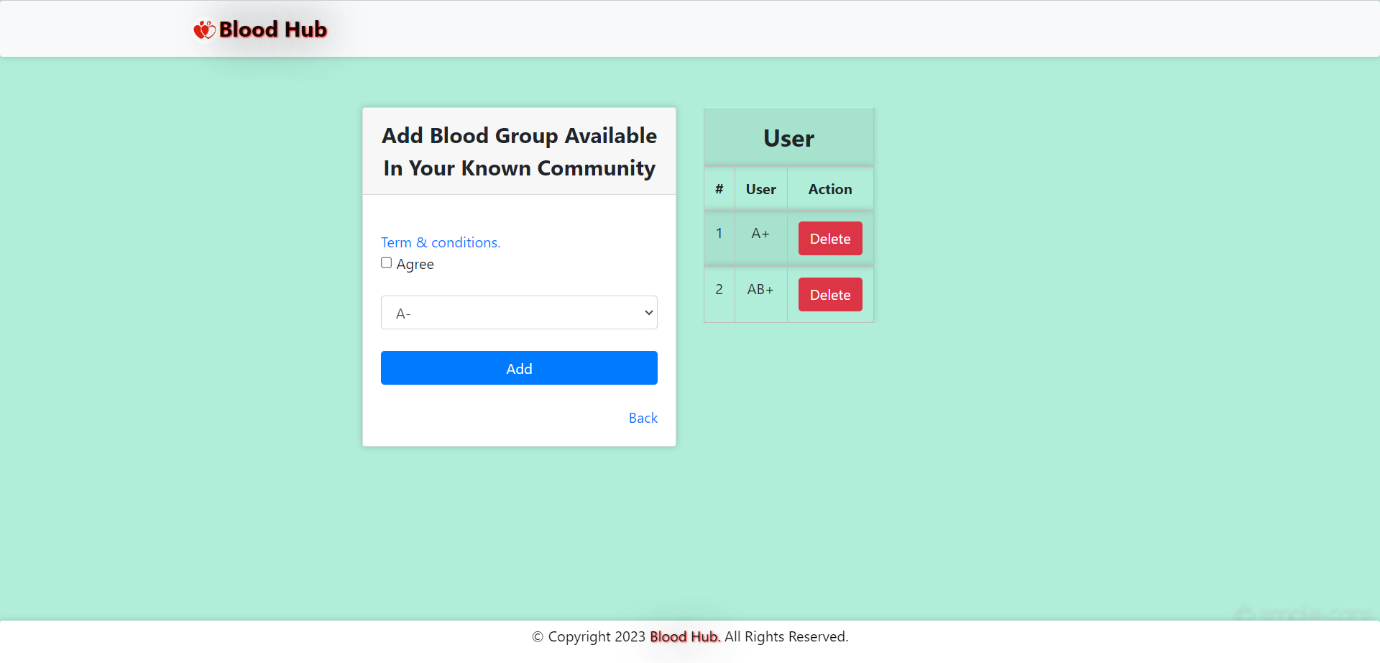
**USER DASHBOARD**

****

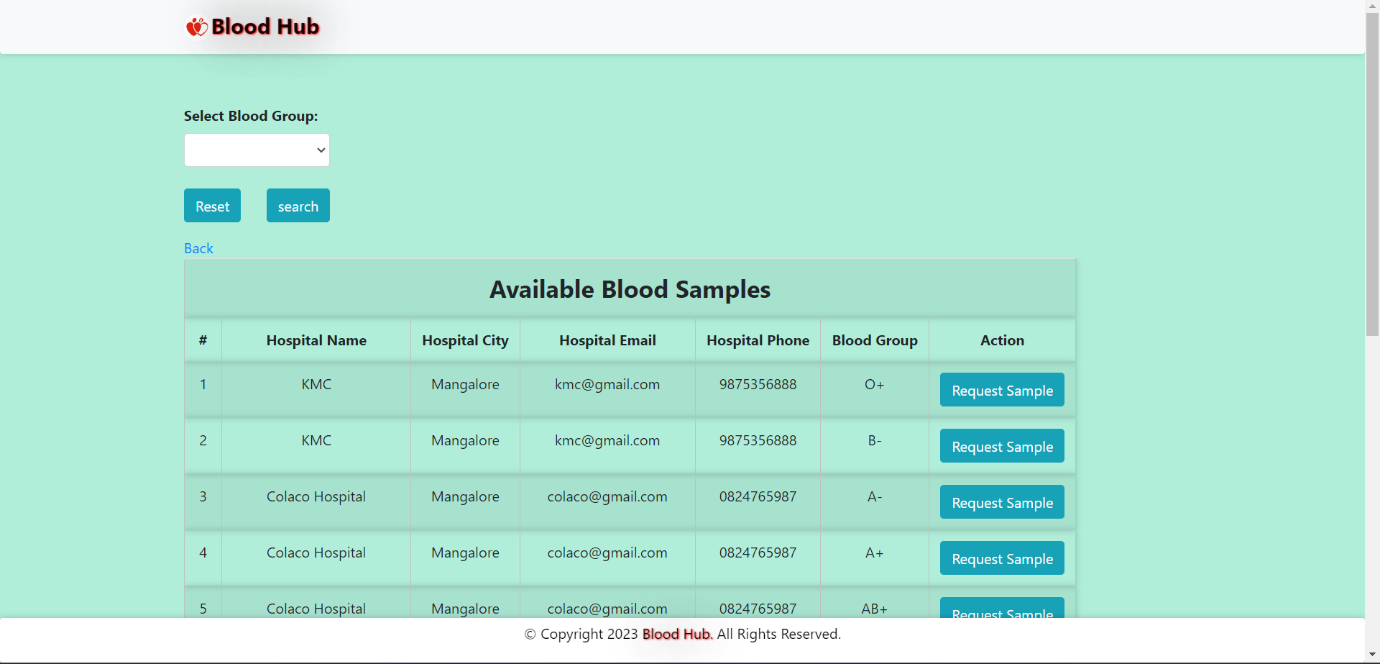
**MY ACCOUNT**

****

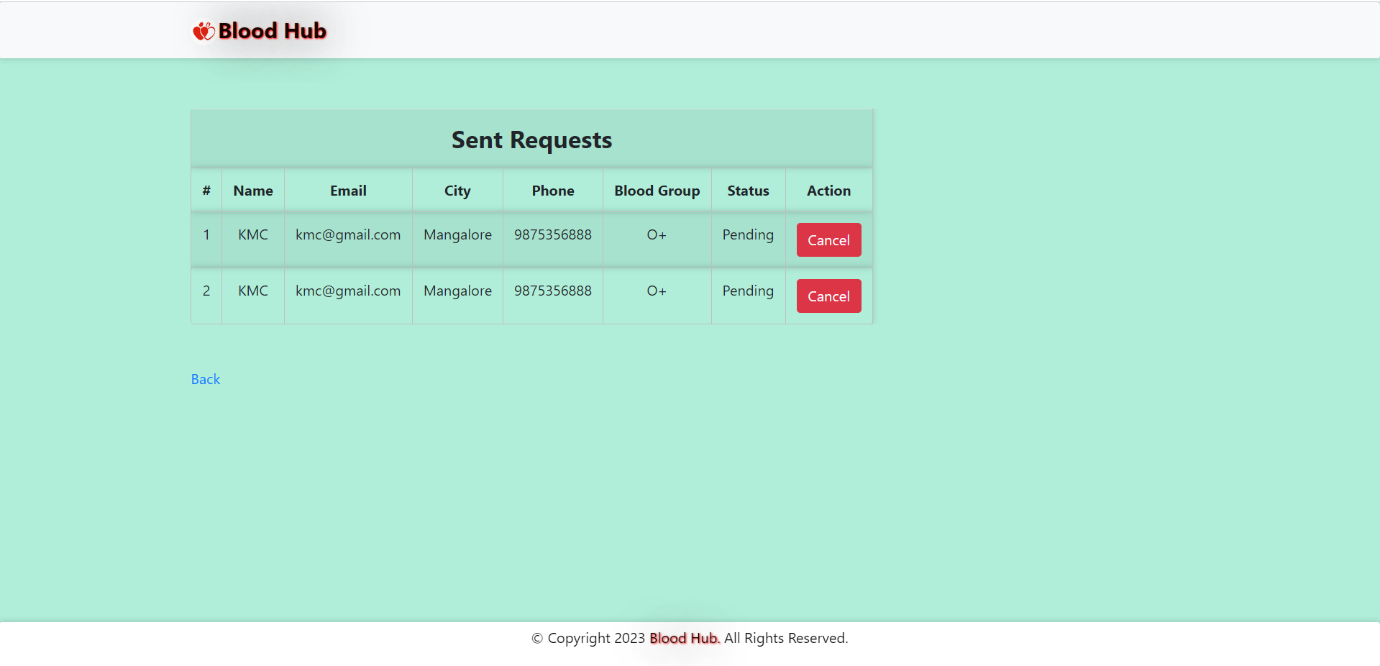
**ADD BLOOD**

****

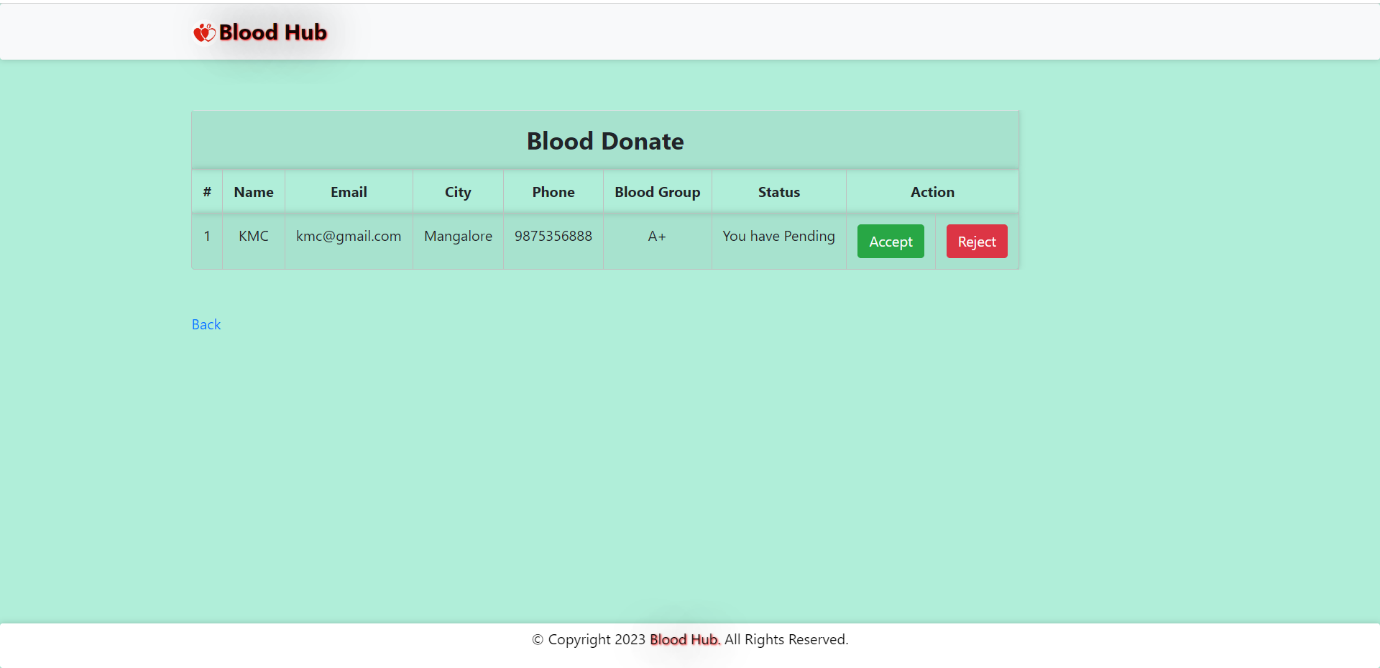
**REQUEST BLOOD**

****

**STATUS OF YOUR REQUEST**

****

**BLOOD DONATION REQUEST**

****

1. **TESTING**

**4.1 Introduction:**

Testing is the major quality control method used during software development. It is the basic function to detect errors. During the requirement analysis and design the output of the document that is usually textual and non-executable after the coding phase the computer programs are available that can be executed for testing purpose. The goal of testing is to uncover requirement, design and coding errors in the program. Testing determines whether the system appears to be working according to the specifications. It is the phase where we try to break the system and we test the system with real scenarios at a point. The implementation is the final and important phase. It involves user-training system testing. In order to ensure successful running of the proposed system, the user tests the system and changes are made according to their needs. The testing involves the testing of the developed system using various kinds of data, while testing errors are noted and correctness is made.

* 1. **Testing Objective**

The objectives of testing are:

* Testing is the process of executing a program with the intent of finding errors.
* A successful test case is one that uncovers un-yet-discovered errors.

System testing is a stage of implementation, which is aimed at ensuring that the system works accurately and efficiently as per the user need, before the live operation commences. As stated before testing is vital to the success of a system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. A series of test are performed before the system is ready for user acceptance test.

**4.3 Test Cases:**

**4.3.1 Hospital**

**Hospital Login page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case id** | **Test Condition** | **Expected Output** | **Result** |
| 1 | If hospital enters invalid email or password | Error Message invalid email or password | success |
| 2 | If hospital enters valid data | Directs to the hospital page | success |

**Hospital Page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case id** | **Test Condition** | **Expected Output** | **Result** |
| 1 | Hospital/Clinic clicks on dashboard | It re-directs to the current dashboard page  Where he can view profile, stock of blood, blood requests, need blood, status of request and logout | success |
| 2 | If hospital clicks on manage profile | He can view and update it | success |
| 3 | If Hospital clicks on stock of blood | He can add or delete blood samples | success |
| 4 | If hospital clicks on blood requests option | He can approve or reject | success |
| 5 | If hospital clicks on need blood | He can search and request | success |
| 6 | If hospital clicks on status of request | He can check whether is approved or not and can cancel. | success |

**4.3.2 User**

**User Login page**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case id** | **Test Condition** | **Expected Output** | **Result** |
| 1 | If user enters invalid email or password | Error Message invalid email or password | success |
| 2 | If user enters valid data | Directs to the hospital page | success |

**User Dashboard**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test case id** | **Test Condition** | **Expected Output** | **Result** |
| 1 | User clicks on dashboard option | It re-directs to the current dashboard page Where he can view his account, blood info, blood available, status of request and blood donation request. | success |
| 2 | If user clicks on Manage profile | He can view and update it | success |
| 3 | If user clicks on blood info | It Directs to the section where he can add or delete samples. | success |
| 4 | If user clicks on blood available | It directs to the section where he can search and request sample | success |
| 5 | If user clicks on status of request | It Directs to the section where he can see whether the request is approved or not and also he can cancel the request. | success |
| 6 | If user clicks on blood donation request | It directs to the page where he can view requests and approve or reject. | success |

1. **SYSTEM SECURITY**

**5.1** **Introduction:**

System security, often referred to as computer or information security, is the practice of protecting computer systems and the information they contain from various threats and vulnerabilities. These threats can come from a variety of sources, including malicious individuals, malware, accidental human errors, and natural disasters. The primary goal of system security is to ensure the confidentiality, integrity, and availability of data and resources within a computer system or network.

Here are some key concepts and components related to system security:

* Confidentiality: This principle ensures that sensitive information remains private and is only accessible to authorized individuals. Confidentiality is typically maintained through access controls, encryption, and user authentication mechanisms.
* Integrity: Integrity ensures that data remains accurate and unaltered during storage, transmission, and processing. It involves methods like data validation, checksums, and digital signatures to detect and prevent unauthorized changes.
* Availability: Availability ensures that the system and its resources are accessible when needed. This involves safeguarding against disruptions, such as hardware failures, DDoS attacks, or natural disasters, which could render the system unavailable.
* Authentication: Authentication verifies the identity of users and entities trying to access a system or network. This can involve methods like username/password authentication, multi-factor authentication (MFA), or biometrics.
* Authorization: Authorization determines what actions and resources a user or entity is allowed to access once their identity is authenticated. Access control lists (ACLs), role-based access control (RBAC), and permissions systems are used to enforce authorization policies.
* Firewalls: Firewalls are security devices or software that monitor and filter incoming and outgoing network traffic based on predefined security rules. They help protect against unauthorized access and network-based attacks.
* Intrusion Detection and Prevention Systems (IDPS): IDPSs monitor network traffic and system activities to identify and respond to suspicious or malicious behavior. They can detect and block intrusion attempts in real-time.
* Antivirus and Antimalware Software: These programs are designed to detect, quarantine, or remove malicious software (malware) from a system. They are essential for protecting against viruses, worms, Trojans, and other malware types.

**5.2 Software Security:**

* There are two user levels in **“Blood Hub”**, Access to the various subsystems will be protected by a user log-in screen that requires a username and password. This gives different views and accessible functions of user levels through the system.
* Email ID once registered to the system cannot be changed to make every user unique and easily identifiable
* Maintaining backups ensures the system database security. The system can be restored in any case of an emergency.
* The server on which the **“Blood Hub”** has to be secured to prevent unauthorized write/delete access. There is no restriction on reading access.
* The proposed website will be secure. There are different categories of users they are Hospitals/Clinics and Users
* Depending upon the category of using the access rights are decided.

1. **CONCLUSION**

In conclusion, the Blood Hub project represents a crucial step forward in modernizing and optimizing the management of blood resources. This documentation has provided a comprehensive overview of the project, from its mission and objectives to its technical aspects and organizational structure. As we wrap up this documentation, here are some key takeaways:

* Lifesaving Mission: The primary goal of the Blood Hub project is to save lives by ensuring the timely availability of safe blood to those in need. This mission is at the core of everything we do.
* Advanced Technology: The project utilizes cutting-edge technology, including robust databases, secure web and mobile applications, and powerful analytics tools, to streamline the blood management process.
* User-Centric Approach: We prioritize the needs of our users, which include hospitals or clinics,, and donors. The system is designed to be intuitive, efficient, and accessible to all stakeholders.
* Sustainability: We are committed to the long-term sustainability of the project through a combination of funding sources, including grants, donations, subscriptions, and support contracts.
* Continuous Improvement: Lifesaver Technologies is dedicated to ongoing improvement and innovation. We will regularly update and enhance the Blood Bank Management System to adapt to changing healthcare needs and technological advancements.

In closing, the Blood Hub is not just a project; it's a lifeline for individuals in need of blood transfusions and a critical tool for healthcare institutions. Lifesaver Technologies remains steadfast in its mission to make a positive impact on healthcare by ensuring a steady supply of safe blood.

1. **FUTURE ENHANCEMNTS**

As technology and healthcare practices continue to evolve, it's important to plan for future enhancements to the Blood Hub project. Here are some potential areas of improvement and expansion for your project documentation:

* Integration with Wearable Devices and IoT: Consider documenting plans to integrate wearable devices and IoT sensors to monitor donors' health and blood quality in real-time. This can enhance donor safety and the quality of collected blood.
* Machine Learning and Predictive Analytics: Discuss the potential for incorporating machine learning and predictive analytics to forecast blood demand, optimize inventory levels, and improve the efficiency of blood distribution.
* Telemedicine Integration: Explore opportunities to integrate telemedicine capabilities into the BBMS. This would enable healthcare providers to connect with donors and patients remotely, making the blood donation and transfusion process more convenient.
* Blockchain for Blood Supply Chain: Document a potential future enhancement involving blockchain technology to create a transparent and tamper-proof ledger of the entire blood supply chain, from donor to recipient.
* Enhanced Mobile Apps: If not already included, outline plans to enhance the BBMS mobile applications, including additional features for donors, such as appointment scheduling, notifications, and gamification to encourage regular blood donations.
* Expansion to New Regions: Describe a strategy for expanding the BBMS to new regions or countries, addressing any regulatory and logistical challenges.
* Artificial Intelligence for Blood Matching: Document plans to use AI algorithms to improve the efficiency and accuracy of blood compatibility testing and cross-matching.
* APIs for Third-Party Integration: Discuss the possibility of developing APIs (Application Programming Interfaces) to allow third-party healthcare systems to integrate with the BBMS, improving interoperability.

1. **WEEKLY PROGRESS REPORTS**

This progress report provides an overview of the key accomplishments, challenges, and future milestones for the Blood Hub project during the reporting period. The project aims to streamline the management of blood resources, enhance accessibility, and save lives. The following sections detail our progress and plans.

**Accomplishments:**

* Project Kick-off and Planning: During this reporting period, we successfully initiated the BBMS project. This included defining project goals, objectives, and establishing the core project team.
* System Design and Architecture: We completed the system design and architecture phase, defining the technical stack, database structure, and user interface design. Key decisions were made regarding technology choices and integration strategies.
* Database Development: The database infrastructure, based on MySQL, has been implemented, providing a solid foundation for data storage and retrieval.
* User Interface Development: We have made significant progress in developing the user interfaces for both the web and mobile applications, ensuring a user-friendly experience.
* Initial User Testing: The project underwent initial user testing to gather feedback and make necessary adjustments to the system's usability and functionality.

**Challenges:**

* Resource Constraints: We encountered some challenges related to resource constraints, including the availability of skilled developers and funding limitations. These challenges have slightly affected our progress in certain areas.

**Upcoming Milestones:**

* Development Continuation: In the upcoming reporting period, we will focus on completing the development phase, including backend functionalities, mobile application development, and user authentication.
* Integration and Testing: We will begin integration testing to ensure seamless communication between different system components and data accuracy.
* Security and Compliance: A significant milestone will be achieving compliance with relevant healthcare regulations and implementing robust security measures.
* Pilot Implementation: Plans for the pilot implementation of the BBMS within select healthcare institutions will be finalized during the next reporting period.
* User Training and Support: We will outline the user training and support strategy, ensuring that stakeholders are well-prepared for the BBMS rollout.

**Conclusion:**

The Blood Hub project has made substantial progress during this reporting period, despite encountering some challenges. We are committed to delivering a robust and user-friendly system that will contribute to the efficient management of blood resources and ultimately save lives.

1. **BIBLIOGRAPHY**

**Books:**

* Core PHP Programming: Using PHP to Build Dynamic Web Sites Book by Leon Atkinson
* High Performance MySQL: Optimization, Backups, Replication, and More Book by Derek J. Balling and Jeremy Zawodny
* Software Engineering by Pankaj Jalote
* PHP Developer's Guide By Greg Buczek
* MYSQL a complete Reference By Alexis Leon, Mathew Leon

**Websites:**

www.tutorialsapoint.com

<https://www.w3schools.com/sql/>

<http://www.stackoverflow.com>

<http://www.Google.com>

http://www.wikipedia.com