High Level Design

Blood Bank Management System

Revision Number: 1.3.2

Last date of revision:11/02/2023

Team 1

Change Record

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Date** | **Author** | **Changes** |
| Revision 1 | 04/01//2023 | Team 1 | Created Basic Program |
| Revision 2 | 06/01/2023 | Team 1 | Implemented Admin Approve functions |
| Revision 3 | 11/01/2023 | Team 1 | Implemented Email sending function |

Contents

# Introduction

* 1. Why this High Level Design Document?
  2. Scope
  3. Definitions
  4. Overview

# General Description

* 1. Product Perspective
  2. Tools used
  3. General Constraints
  4. Assumptions
  5. Special Design aspects

# Design Details

* 1. Main Design Features
  2. Application Architecture
  3. Technology Architecture
     1. Web Application Architecture
     2. Presentation Layer
     3. Data Access Layer
     4. Tools Used
  4. Standards
  5. Database design
  6. Files
  7. User Interface
  8. Reports
  9. Error Handling
  10. Interfaces
  11. Help
  12. Performance
  13. Security
  14. Reliability
  15. Maintainability
  16. Portability
  17. Reusability
  18. Application compatibility
  19. Resource utilization
  20. Major Classes

# Introduction

* 1. Why this High Level Design Document?

The purpose of this High Level Design (HLD) Document is to add the necessary detail to the current project description to represent a suitable model for coding. This document is also intended to help detect contradictions prior to coding, and can be used as a reference manual for how the modules interact at a high level.

* 1. Scope

The HLD documentation presents the structure of the system, such as the database architecture, application architecture (layers), application flow (Navigation), and technology architecture. The HLD uses non-technical to mildly-technical terms which should be understandable to the administrators of the system.

* 1. Definitions
     + BBMS– Blood Bank Management System.
     + Firewall –Functionality that can allow or block certain ports and addresses.
     + ER – Entity Relation Diagram
     + JDBC – A possible Java-based interface between IPTables and the Database.
     + Tomcat – a free, open-source implementation of Java Servlet and JavaServer Pages technologies developed under the Jakarta project at the Apache Software Foundation.
     + Apache - An open source Web server.
     + ER – Entity Relation Diagram
     + API – Application Programming Interface.
     + Kernel – Core of an operating system, a kernel manages the machine’s hardware resources (including the processor and the memory), and provides and controls the way any other software component can access these resources.
     + DHCP – (Dynamic Host Configuration Protocol) – This is a protocol that lets network administrators centrally manage and automate the assignment of IP Addresses on the corporate network.
     + Gateway –Bridges the gap between the internet and a local network.
  2. Overview

The HLD will:

* + - present all of the design aspects and define them in detail
    - describe the user interface being implemented
    - describe the hardware and software interfaces
    - describe the performance requirements
    - include design features and the architecture of the project
    - list and describe the non-functional attributes like:
      * security
      * reliability
      * maintainability
      * portability
      * reusability
      * application compatibility
      * resource utilization
      * serviceability

# General Description

* 1. Product Perspective

Blood Bank Management Application is a Web-based application used by Donors, Receivers,hospitals and blood center. This system can manage profile, manage blood bank staff view blood donation records view information about blood donation. This system have function use to keep the donation record of the donor. Besides, the process of adding data, delete data, update the data can be done anywhere by accessing the application.

* 1. Tools used

1. Jude, a Java based UML design program, is used to generate all of the diagrams used in analysis and design phases of the project.
2. The project will have a relational database backend that is SQL based. The actual software used is MySQL.
3. Interfacing with the database to display information on the user’s web browser will be done using JPA. The Java Persistence API (JPA) is a specification of Java. It is used to persist data between Java object and relational database. JPA acts as a bridge between object-oriented domain models and relational database systems.
4. Spring Boot is an open source Java-based framework used to build stand-alone and production ready spring applications.
5. MySQL Workbench is a unified visual tool for database architects. It provides data modeling, SQL development for the programmer.
6. My Structured Query Language (MySQL) database management system is used to design the prototype database.
7. To model the analysis and design of the proposed system Unified Modeling Language (UML) modeling techniques is used and Hyper-Text Transfer Protocol (HTML) is used to develop the system prototype.
8. Tomcat compiles JSP pages into servlets to be displayed through Apache..
9. Apache An open source web server that will display requested pages
10. Automated interfacing with the database behind the scenes will be JDBC.
11. DHCP assigns IP addresses.
    1. General Constraints

The Blood Bank Management Application only cover English language. It does not support other languages. This application needs human intervention to update profiles and approve donors. Administrators should not be required to do anything besides the initial setup, and users should not be required to know any of the workings. Without logging in, the user will only have the ability to be considered as a donor or as a blood seeker. Logging in and being registered is must for User .

* 1. Assumptions

This project is based on the idea making Blood Bank information easily accessible, and the goal is to make this idea a reality using Software Engineering practices. In doing so, many documents are created, and it is assumed that design flaws will be found early on. It is also assumed that all aspects of this project have the ability to work together in the way the designer is expecting.

Another assumption is that the current intended documentation will suffice to make this project count towards the Software Engineering Subtract. There is also an assumption that none of the work or hardware will be stolen or sabotaged. The final assumption is that a demonstration and presentation will be possible by the end.

* 1. Special Design aspects

This application is specified to be used by the Donors who wants to donate blood and Receivers who are in need of blood. When the user registers himself or herself he/she will be sent a mail regarding their registration information and the details of user and seeker should be approved even after registration .

# Design Details

* 1. Main Design Features

The main design features include five major parts: the architecture, the user interface design, external interface, the database, process relation, and automation. In order to make these designs easier to understand, the design has been illustrated in attached diagrams (ER, Use Case, and Screen Shots).

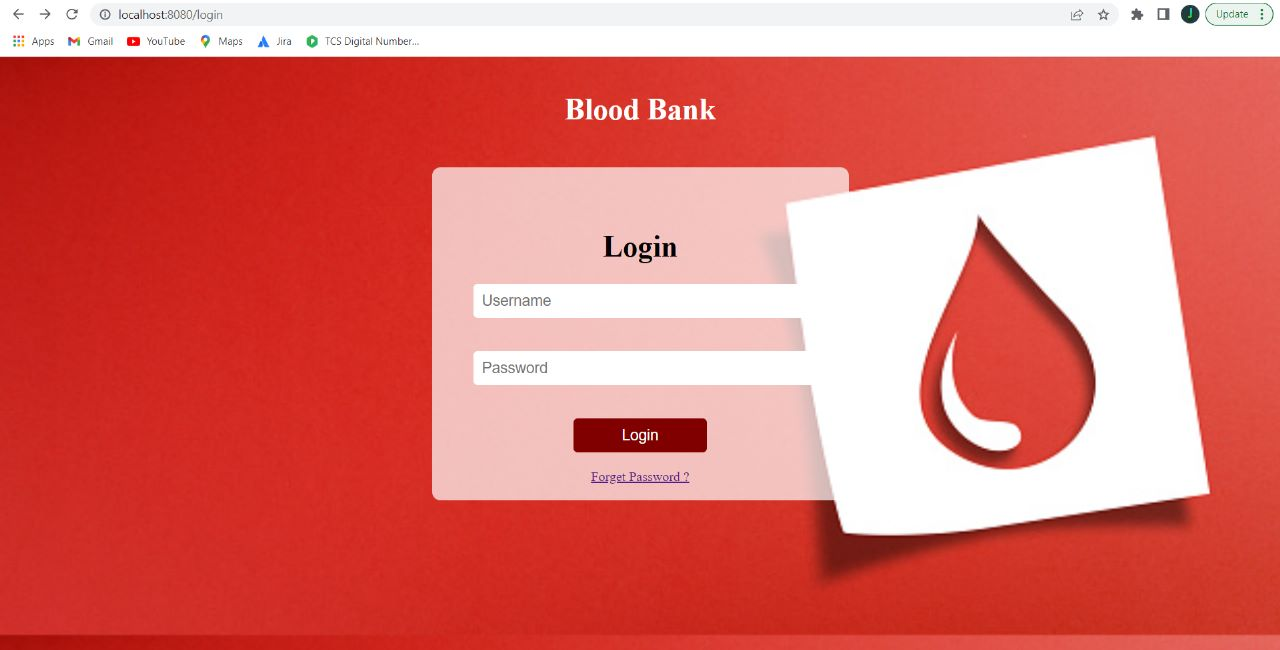
Screen Shot Breakdown:

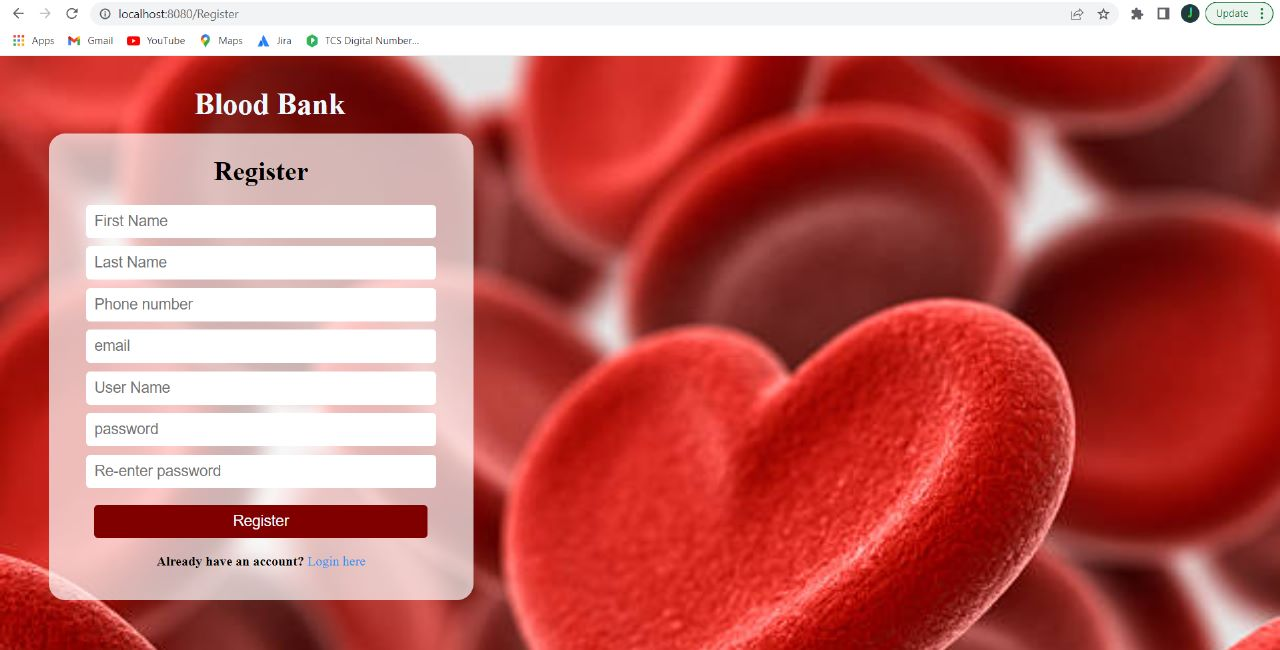
Index



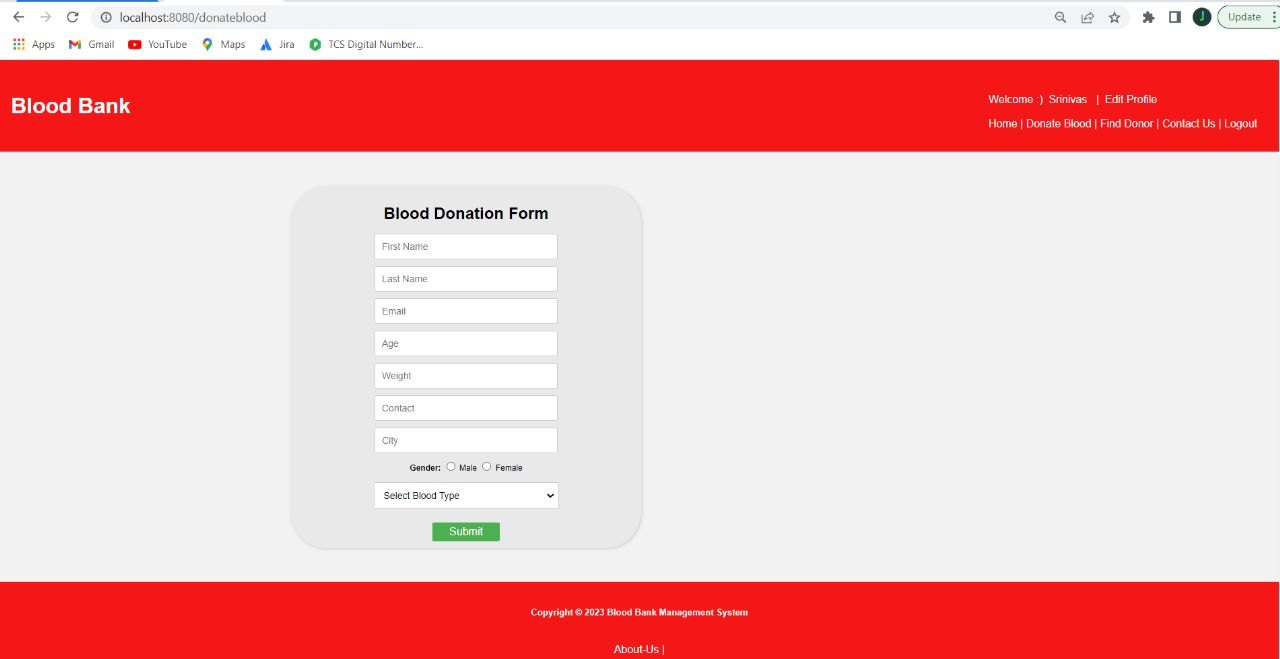
User:

login page

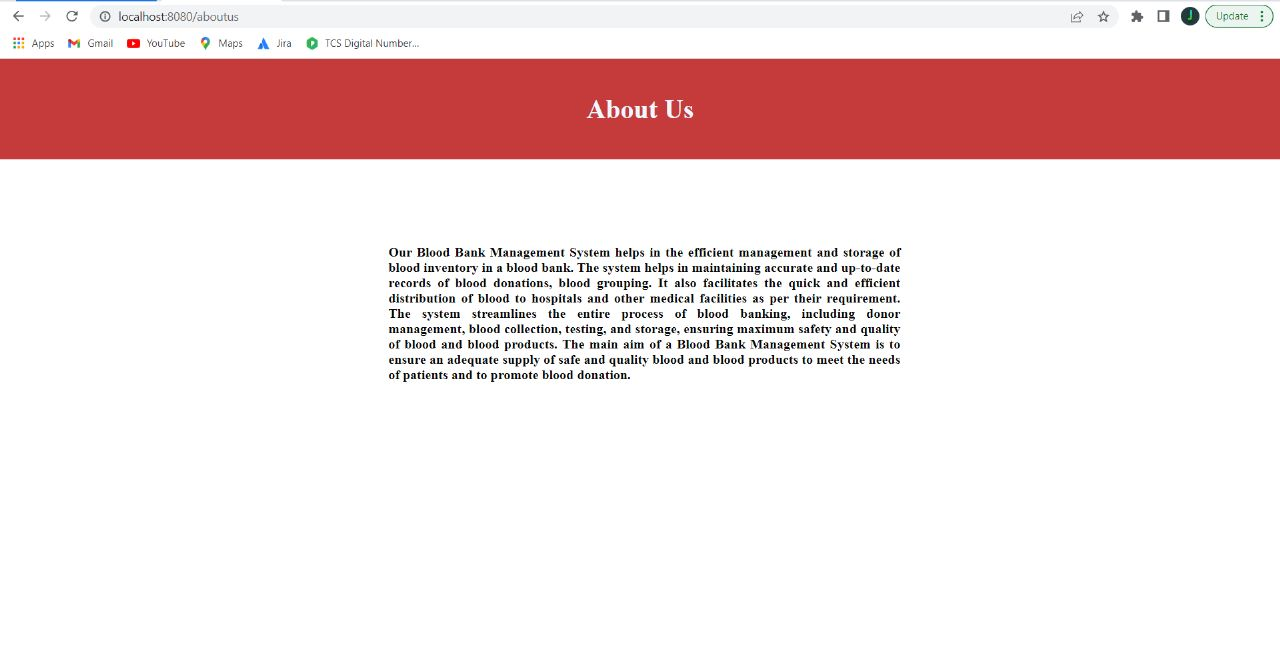


Register:

Donor:

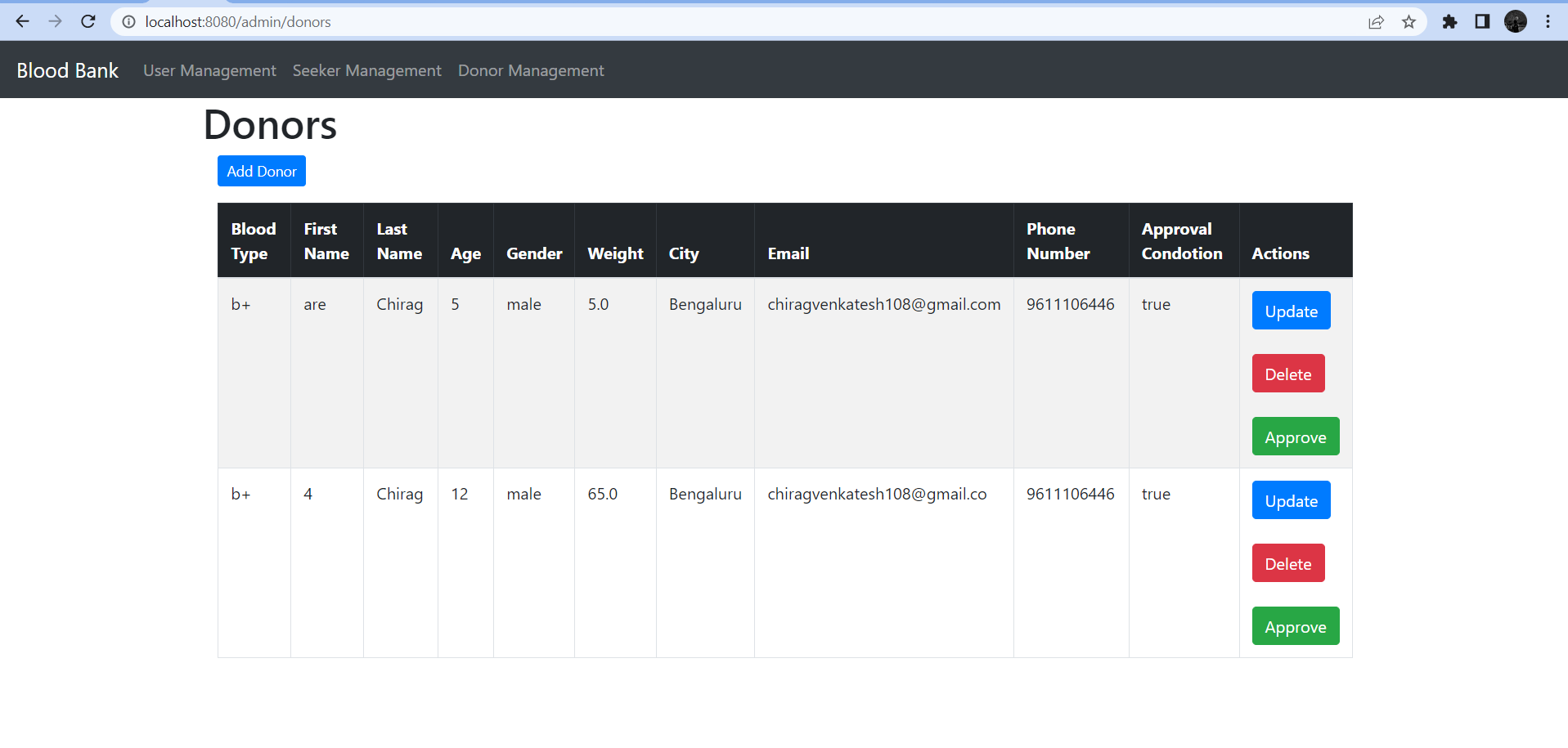


About us:

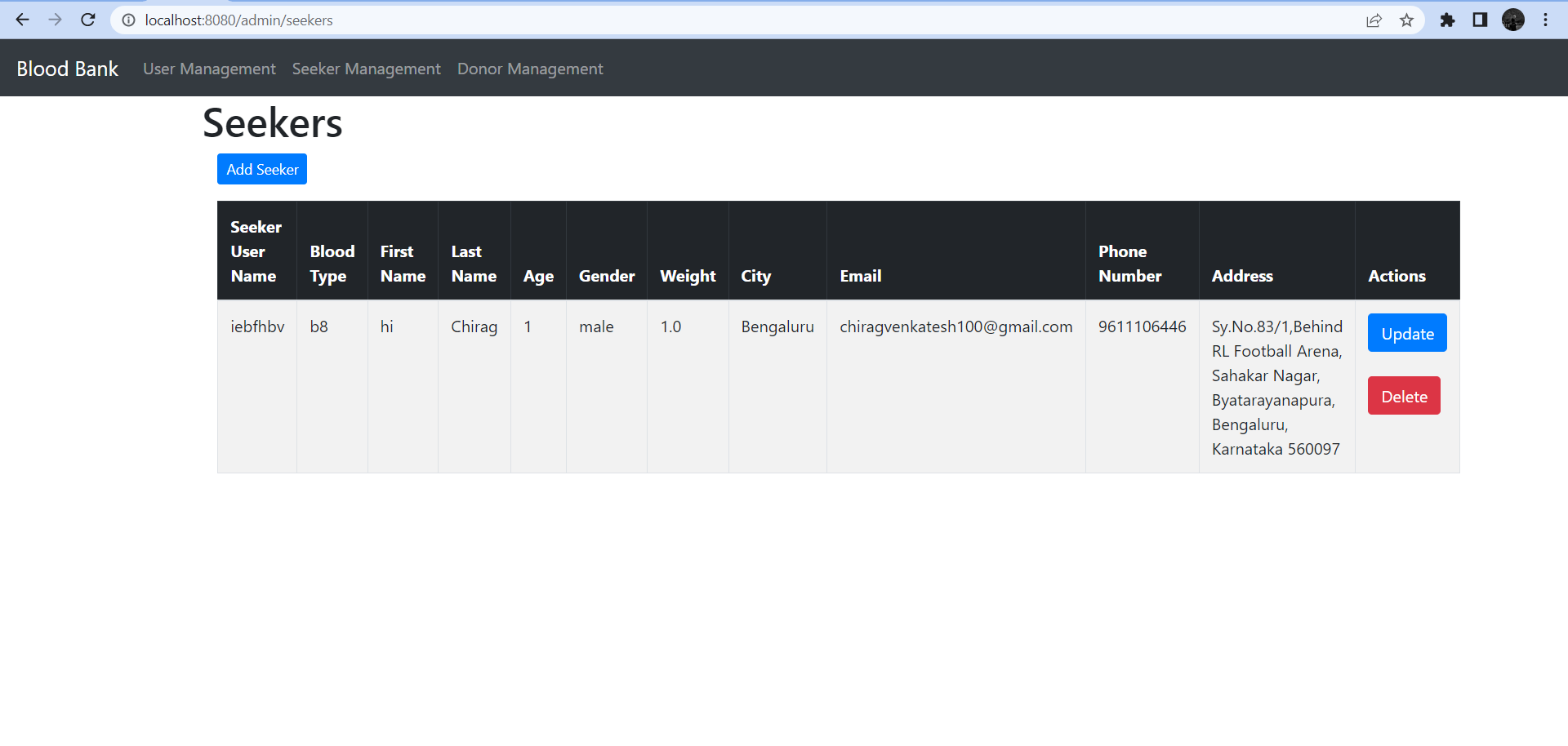


Administrator:

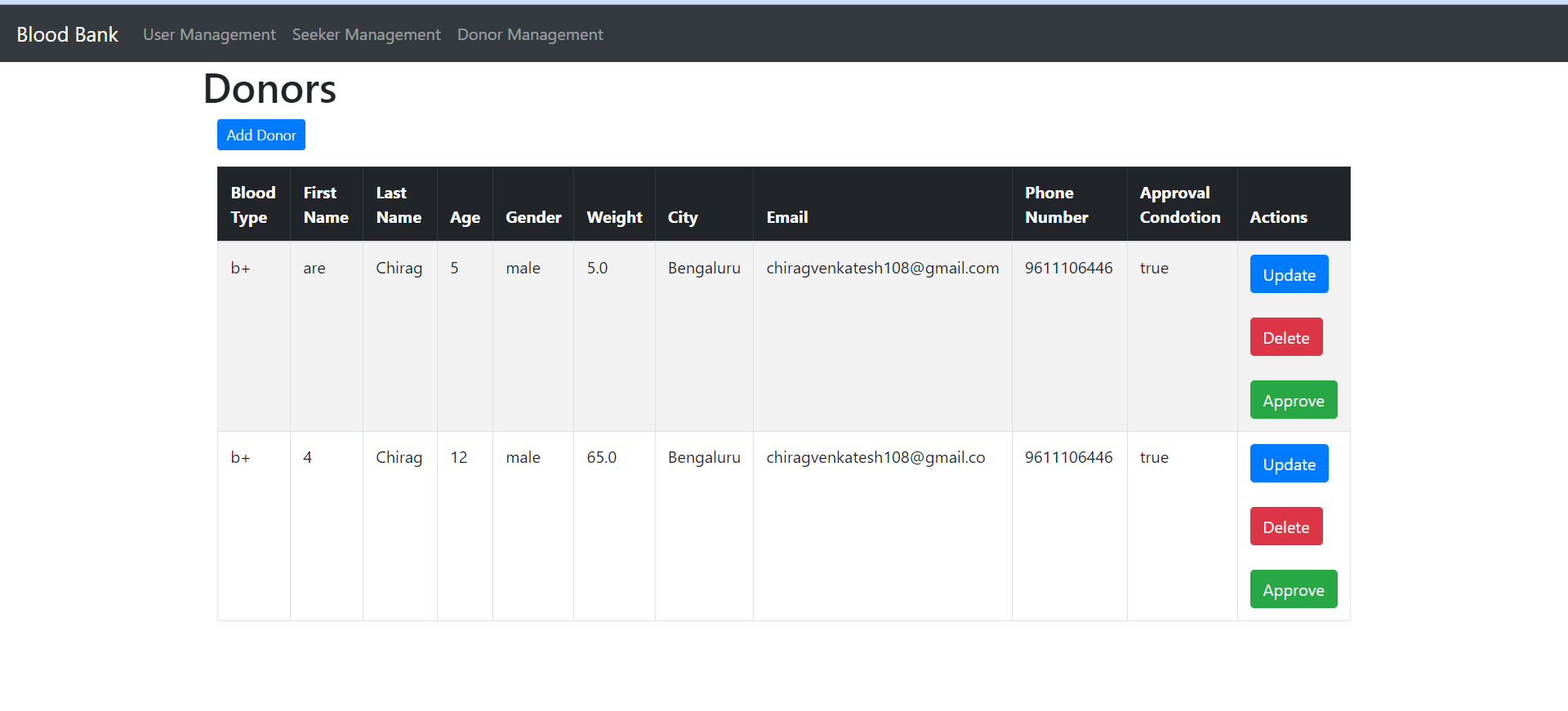
User Info Page:



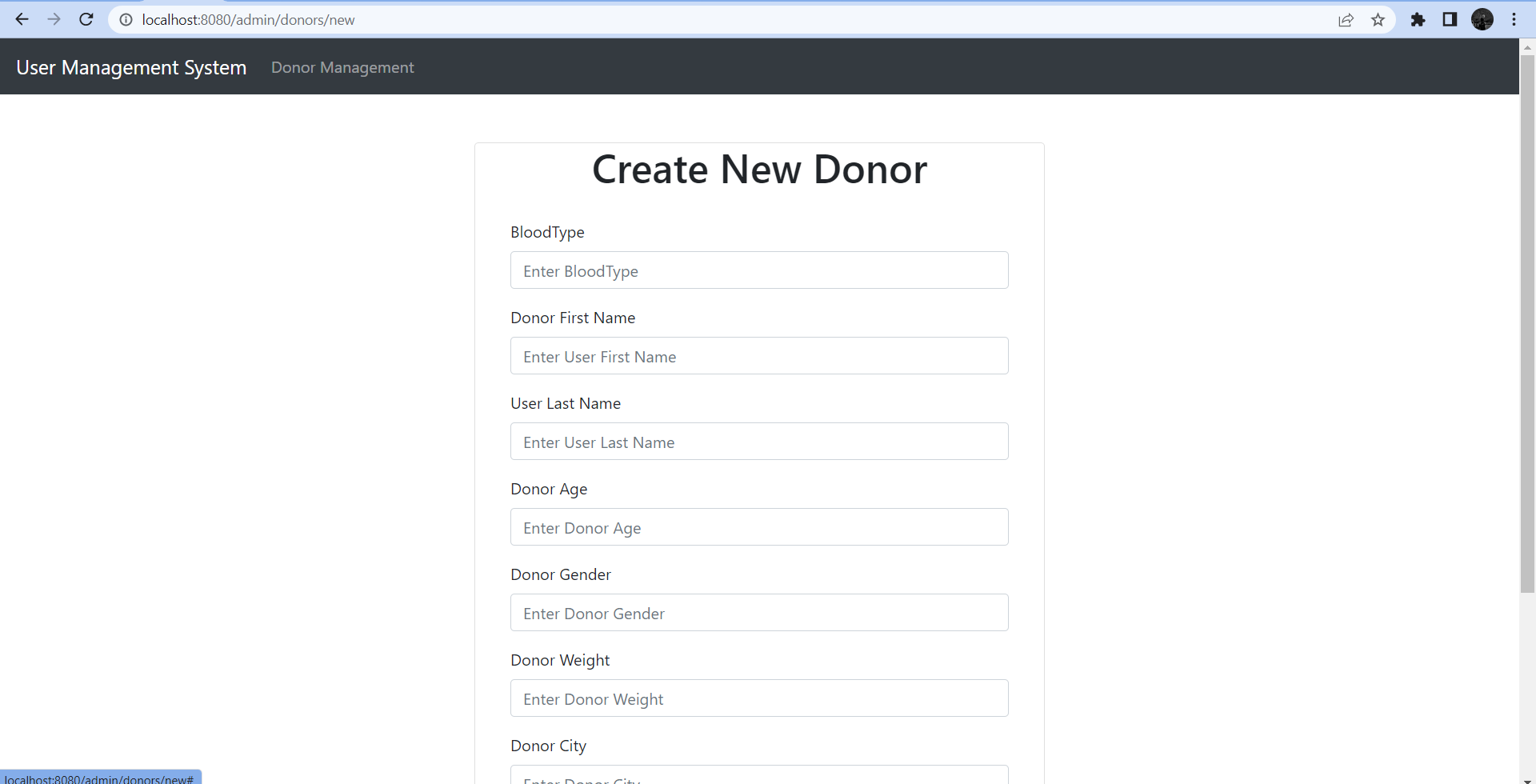
Seeker Info Page:



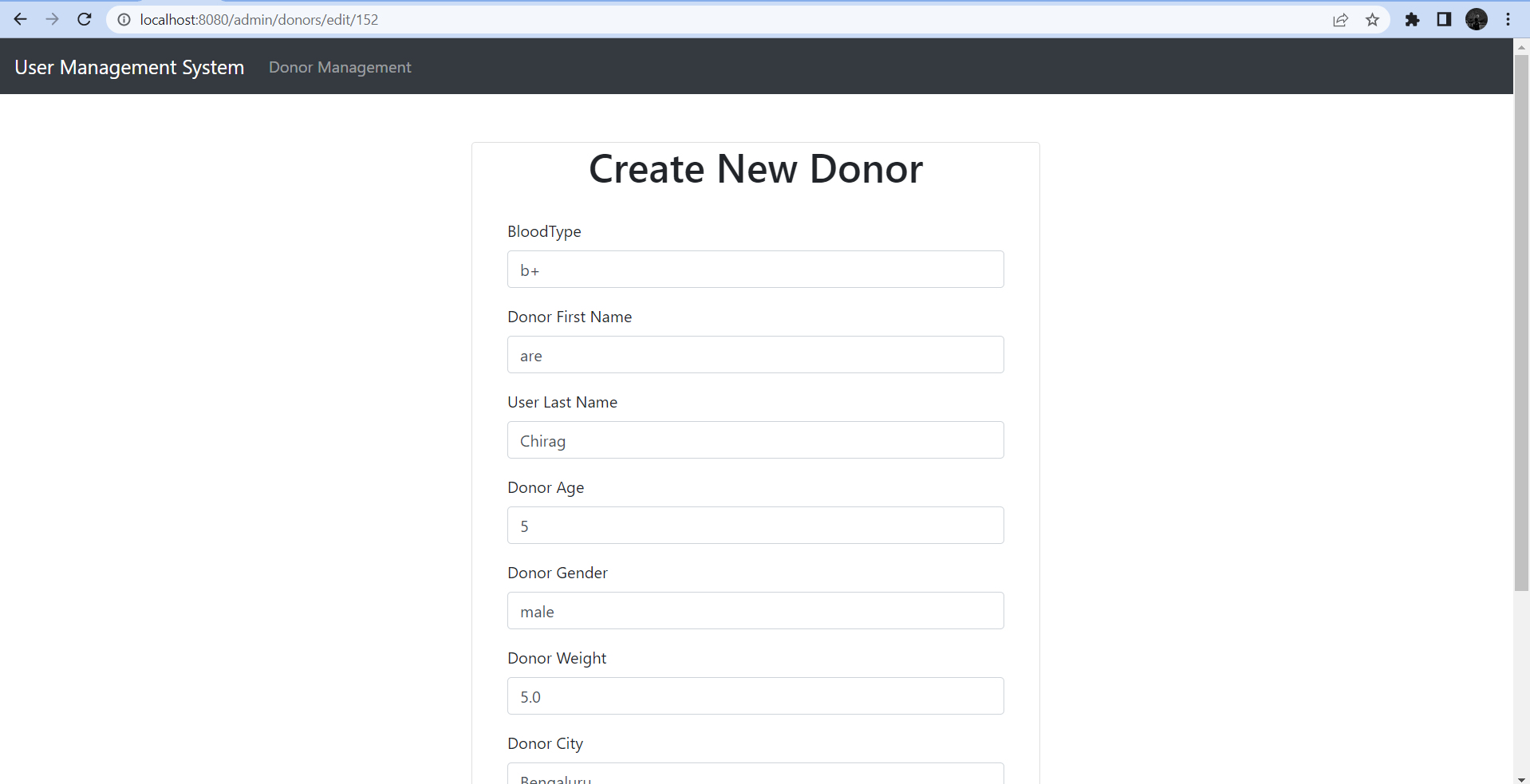
Donor Info Page:



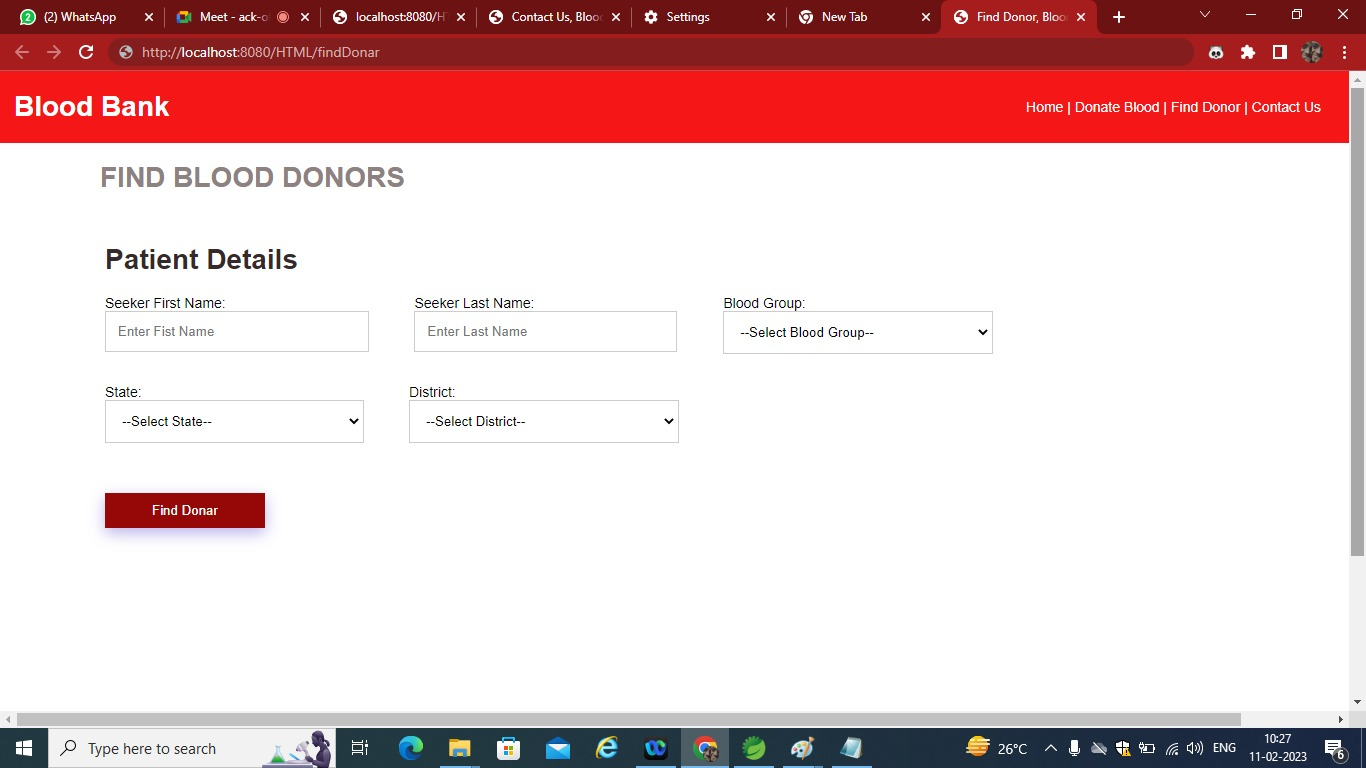
Donor Add:



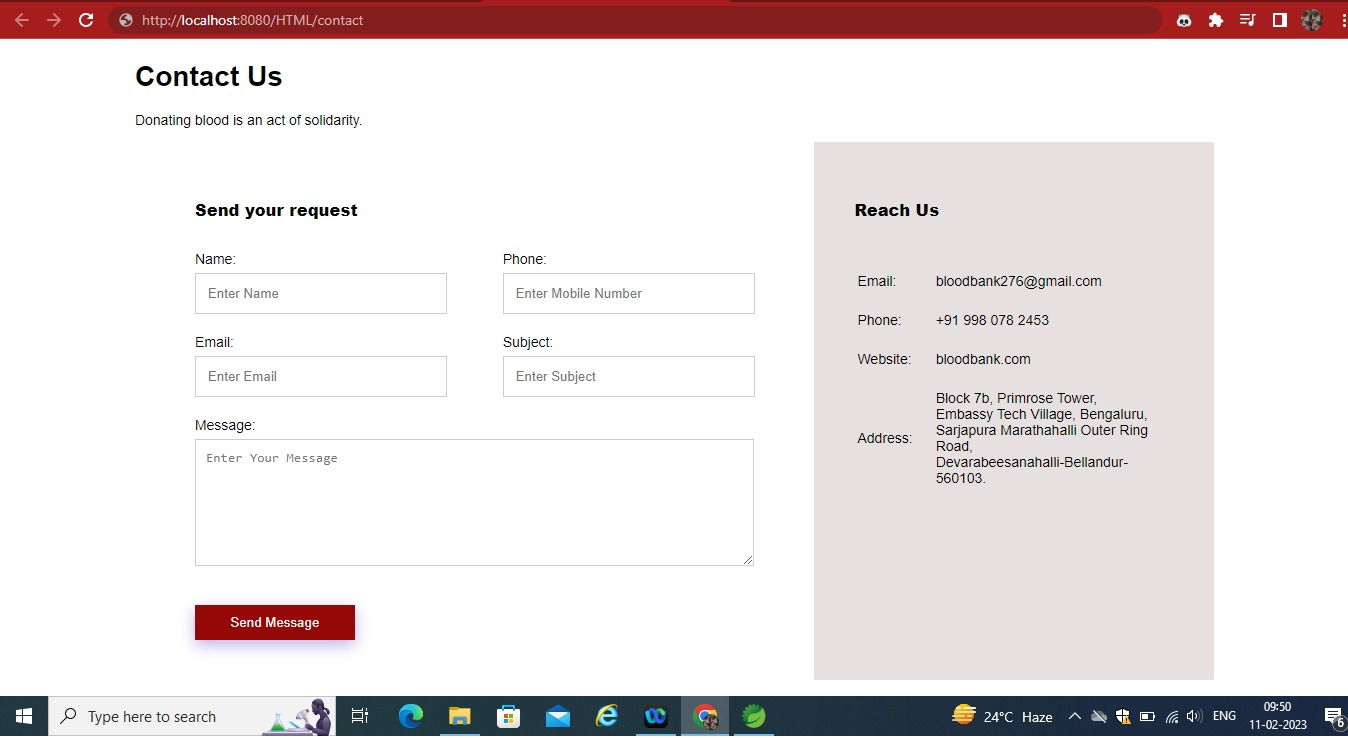
Donor Update:



Receiver:

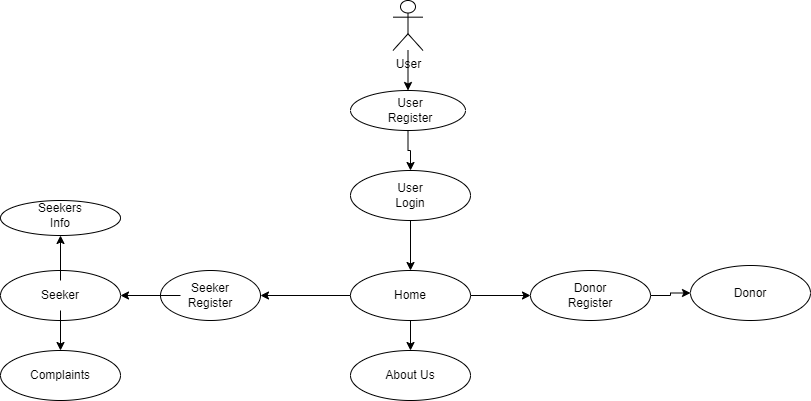


Aboout Us:

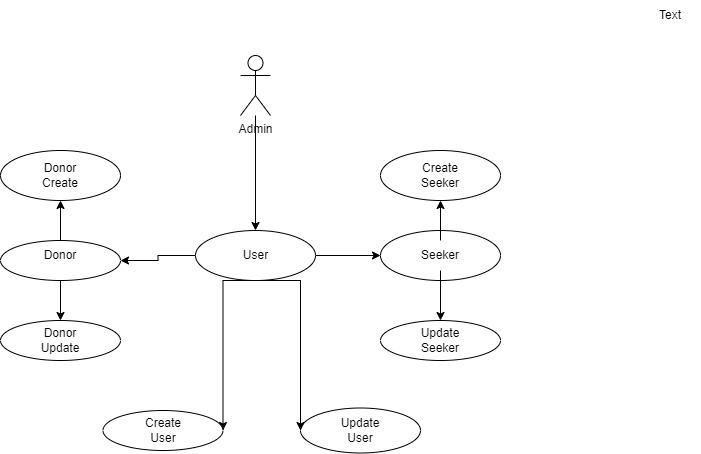


* 1. Application Architecture

User:



Admin Architecture:

* 1. 
  2. Technology
     1. Web Application Architecture

The front end of the program is a web application. Functionality will vary based user privileges if a user is logged in. Normal users are required to log in,to view their personal and other information. Administrators will have access administrative abilities to create delete update user donor and seeker.

* + 1. Presentation Layer

Presentation Layer will contain the information pertaining to the access for Donors and the Receivers. The donor will have the access to log in the system and modify his/her personal details and can update their blood type details. The receivers have the access to log in the system and search for his or her blood type.

* + 1. Data Access Layer

The database will be accessible to all users, administrators, and automated services. A login will determine what parts of the database can be accessed and changed.

* + 1. Tools Used

See section 2.2 for tools used in the design of this project.

* 1. Standards

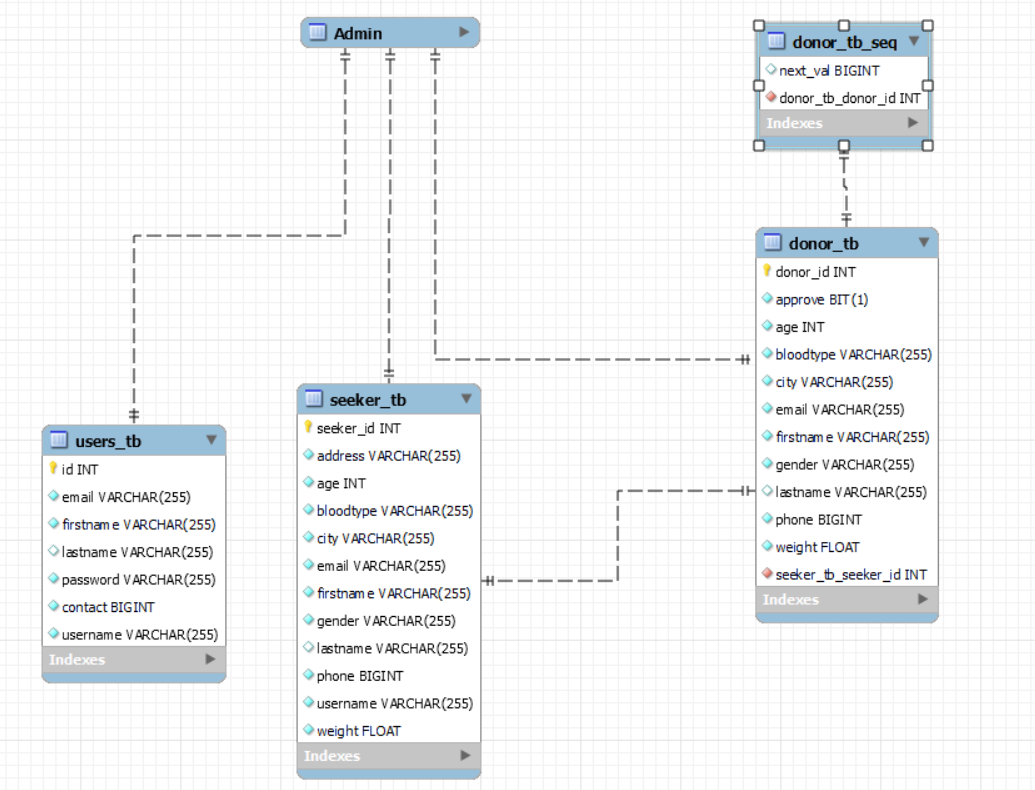
Database – relational

Inputs – entered through text field and stored in database.

Security – username and password are required for access to the system.

Quality – by keeping the interface simple and direct, quality should be kept at a maximum.

* 1. Database design



* 1. Files

This product will not use a large number of files. Tomcat uses JSP pages. A file will be used to store all of the usernames and passwords and all other attributes specified for those users. This file will be accessed at login. It can be modified by the administrator at any time. Another file will store the database of donors and another for seekers.This file can also be updated by admin at anytime

* 1. User Interface

The user interface is a very simple plain layout with little to no graphics. It will display information very clearly for the user and will primarily output information to the user through HTML pages using Bootstrap, Themeleafy ,CSS. Administrative screens are use mainly for input through text fields in Themeleafy pages.

* 1. Reports

The reports will display the Transfer of blood and its type between the Donor and the receiver. Reports can be accessed by the donor to know about his/her blood transfer.

* 1. Error Handling

Should errors be encountered, an explanation will be displayed as to what went wrong. An error will be defined as anything that falls outside the normal and intended usage.

* 1. Interfaces

There are four main interfaces for this project. First, the donor interface, which consists of the login page action and then the interface related to his information where he/she can update their information and blood group which he/she can donate. Second is the interface for receiver where the receiver can update his/her informationand and specify the type of blood they are looking for. The third interface is the user interface Tomcat. The final interface will be admin interface.

* 1. Help

Help will come in the form of all the documentation created prior to coding, which explain the intended uses. Should time allow, detailed instructions will be written on how to create and implement the system with the intentions of publishing as an Open Source solution.

* 1. Performance

Performance is going to be very important for this project. For everything to run smoothly for this project, the gateways will have to be able to update data on the database and refresh the donors database before it is supposed to do so again. This is likely to be the most processor intensive aspect of the project. The gateways will also need to supply requested pages to the users at a reasonable speed. The database server will need to keep up with all database requests and transactions.

* 1. Security

Because security is not the prime focus of this project, only the minimal aspects of security will be implemented. A username and password will be required to log into an

donor and receiver interface and database. For now, all data will be sent in plain text.

Verification of user phone number is also outside the scope of this project. There will also be failed attempts of an donor or receiver logging in if entered information is not correct.

* 1. Reliability

A redundant database server will be implemented so that if the main database server stops responding, the gateways will automatically start using the other server. The mechanism used for syncing these two databases has not yet been fully established. Likely candidate solutions include:

* + - Each user if updated will be updated at both interfaces.
    - An archive field being implemented and the server constantly searching the tables for new data.
    - A full periodic refresh for the entire database.
    - A trigger being used on the main database where all data is automatically stored even updated in different interfaces.
  1. Maintainability

Very little maintenance should be required for this setup. An initial configuration will be the only system required interaction after system is put together. The only other user maintenance would be any changes to settings after setup, and any specified special cases where user settings or history need to be changed. Physical maintenance on the system’s parts may be required, and would result in temporary loss of data or Internet. Upgrades of hardware and software should have little effect on this project, but may result in downtime.

* 1. Portability

This system should have the ability that, once it is together, the entire system should be able to be physically moved to any location. Code and program portability should be possible between kernel-recompiled Linux distributions. For everything to work properly, all components should be compiled from source.

* 1. Reusability

The code written and the components used should have the ability to be reused with no problems. Should time allow, and detailed instructions are written on how to create this project, everything will be completely reusable to anyone.

* 1. Application compatibility

The different components for this project will be using Java as an interface between them. Each component will have its own task to perform, and it is the job of the Java code to ensure proper transfer of information.

* 1. Resource utilization

When any task is performed, it will likely use all the processing power available until that function is finished. The gateways are likely to use their processors the hardest when they are refreshing donors or seekers information, but this will depend largely on how many on how many users are registered as donors and receivers.

* 1. Major Classes

There are a total of four major classes: Donors, Users, R e c i v e r s , ,A d m i n. The relationships between these major classes are:

A User has to register and login to get to the home page where he can decide to be a donor receiver.

A Donor can Update his/her information and can also update his/her BloodType.

A Seeker can Update his/her information and search for his BloodType if available.

Admin can create,delete and update the database of donors,users,seekers and should also approve donor information.