A PROJECT REPORT ON

BLOOD-BANK MANAGEMENT

A Project report submitted in partial fulfilment of the requirement for
The award of the Degree of
MASTER OF COMPUTER APPLICATION

Submitted By:
Snehal Prusty
(2224100019)

Under Guidance of:
Mr. Manjit Kumar Nayak
(Assistant Prof.,School of Computer Science)

Department of Computer Science and Application

ODISHA UNIVERSITY OF TECHNOLOGY AND RESEARCH

Techno Campus, Ghatikia, Mahalaxmi Vihar, Bhubaneswar-751029
(Academic Year 2023-24)

ODISHA UNIVERSITY OF TECHNOLOGY AND RESEARCH

Techno Campus, Ghatikia, Mahalaxmi Vihar, Bhubaneswar-751029

Department of Computer Science and Application

CERTIFICATE

This is to certify that the project report entitled **"BLOOD-BANK MANAGEMENT"** being submitted by **Snehal Prusty** bearing the registration no: **2224100019** partial fulfillment for the award of the Degree of Master of Technology in Computer Science and Application to the Odisha University of Technology and Research is a record of bonafide work carried out by him under my guidance and supervision.

The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Signature of Internal Guide	**Signature of Head of the department**
Mr. M.K.Nayak	Dr. jibitesh Mishra
(Ass Professor,	Head of School
School of Computer Science)	School of Computer Science

ODISHA UNIVERSITY OF TECHNOLOGY AND RESEARCH

Techno Campus, Ghatikia, Mahalaxmi Vihar, Bhubaneswar-751029

Department of Computer Science and Application

DECLARATION

Snehal Prusty bearing Registration No: **2224100019,** a bonafide student of **Odisha University of Technology and Research,** would like to declare that the project titled **"BLOOD-BANK MANAGEMENT"** A partial fulfilment of MCA Degree course of Odisha University of Technology and Research is my original work in the year 2023 Under the guidance of **Mr. Manjit Kumar Nayak** Department of Computer Science and Application and it has not previously formed the basis for any degree or diploma or other any similar title submitted to any university.

Date: **Snehal Prusty**

OUTR,BBSR **(2224100019)**

IDENTIFICATION

NAME	:	**Snehal Prusty**
ROLL NO	:	**2224100019**
PROJECT TITLE	:	**BLOOD-BANK MANAGEMENT**
COMMENCING DATE	:	10.11.2023
SUBMISSION DATE	:	05.01.2024

APPROVED BY

HEAD OF THE

SCHOOL OF
COMPUTER SCIENCE

ACKNOWLEGEMENT

What is written or mentioned in this sheet will hardly be adequate in return for the amount of help and co-operation we have received from all of the people who have contributed to make this project a reality. But we will still try our level best to thank them.

First of all we wish to express our sincere thanks to **Mr. Manjit Kumar Nayak** , our project guide in this undertaking, who has always there for us to offer help in all possible ways, ready to solve any problem with a smiling face. We own him a lot due to boost that he provided us which further allowed the successful completion of the project within the stipulated time.

Finally we wish to express our heartiest thanks to our parents, all our teachers and friends who played a very important part in making our project complete and successful.

Thanking You!
Yours's Truly

Snehal Prusty
(2224100019)

Abstraction

The project report on “ BLOOD-BANK MANAGEMENT “ is a part of MCA course provided by Odisha University Technology and Research, Bhubaneswar. The project is designed according to the requirements and its contents, details and other reports.

We have made every effort to rectify the errors from the project reports and we hope that the reports are suitable enough to be presented.

CONTENT

CHAPTER-1

INTRODUCTION

1.1 PROBLEM DEFINITION

Manual blood bank systems can face various challenges that can affect their efficiency and accuracy. Some common problems include:

Data Inaccuracy: Manual data entry is prone to errors, leading to inaccuracies in donor and recipient information, blood typing, and inventory levels.

Time-Consuming: Managing records, tracking inventory, and coordinating blood donations manually can be time-consuming, making it less efficient than automated systems.

Limited Accessibility: Paper-based systems may restrict access to information, making it difficult for different departments or locations to share real-time data.

Inventory Management Issues: Keeping track of blood inventory levels, expiration dates, and matching blood types manually can result in shortages, wastage, or mismatches.

Difficult Communication: Coordinating with donors, recipients, and medical staff manually can be challenging, leading to delays and miscommunication.

Security Concerns: Paper-based systems may lack adequate security measures, making sensitive donor and patient information vulnerable to unauthorized access.

Reporting Challenges: Generating reports and analyzing trends may be cumbersome, hindering the ability to make data-driven decisions for optimizing blood bank operations.

Risk of Loss: Paper records are susceptible to damage, loss, or theft, which can lead to critical information being compromised.

Limited Scalability: As the demand for blood services grows, manual systems may struggle to scale up to meet the increasing needs efficiently.

Audit Trail Difficulties: Tracking changes, updates, and access to records can be challenging in manual systems, making it difficult to maintain a comprehensive audit trail.

Automating blood bank systems can address many of these issues, improving overall efficiency, accuracy, and security.

1.2 ABOUT PROJECT

This project aims to develop a robust Blood Bank Management System utilizing the MERN (MongoDB, Express.js, React.js, Node.js) stack. The system focuses on streamlining the entire blood donation and distribution process, ensuring efficient management of donor records, real-time inventory tracking, and seamless communication between stakeholders. The platform integrates user-friendly interfaces for donors, administrators, and healthcare professionals, enhancing

accessibility and usability.
Key features include a dynamic donor registration system, an intelligent inventory management module, and a responsive dashboard for administrators to monitor and analyze blood supply levels. The incorporation of the MERN stack enables scalability, flexibility, and rapid development, ensuring adaptability to evolving technological landscapes.
The project emphasizes data security and confidentiality by implementing encryption protocols and user authentication mechanisms. Additionally, the application explores the integration of SMS or email notifications for donors, encouraging regular participation and fostering a sense of community engagement.

Through this MERN-based Blood Bank Management System, the project aims to contribute to the optimization of blood bank operations, fostering a more resilient and responsive healthcare system. The proposed solution aspires to be a scalable model for blood banks, combining technological innovation with a user-centric approach to positively impact blood donation and distribution practices.

Here we see multiple Roles that is **Doner, Admin, Hospital , Organisation**

Importance of Blood Banks:

Blood banks play a crucial role in healthcare by collecting, testing, storing, and distributing blood and blood products for medical treatments and emergencies.
Blood Donation:

Blood banks rely on voluntary blood donations from individuals. Regular and voluntary donors help ensure a steady and safe blood supply.
Blood Types:

Blood is categorized into different types based on the presence or absence of antigens and antibodies. The main blood types are A, B, AB, and O, with Rh-positive or Rh-negative factors.
Compatibility Matching:

Blood banks carefully match donor blood with the recipient's blood type to prevent transfusion reactions. ABO and Rh compatibility are crucial factors in this process.
Screening and Testing:

Donated blood undergoes rigorous screening and testing for infectious diseases, ensuring that the blood supply remains safe for transfusions.
Blood Components:

Blood can be separated into various components, such as red blood cells, plasma, platelets, and cryoprecipitate. This allows for targeted treatments and maximizes the use of each donation.
Emergency Response:

Blood banks play a vital role in disaster response and emergency situations, providing life-saving blood transfusions to those in critical need.
Storage and Shelf Life:

Blood and its components have specific storage requirements, and blood banks must carefully manage inventory to prevent wastage and ensure the availability of fresh products.
Volunteerism and Community Engagement:

Blood donation campaigns and drives are often organized to encourage community participation and raise awareness about the importance of donating blood.
Global Challenges:

Some regions face challenges in maintaining an adequate and safe blood supply due to factors like population growth, limited resources, and cultural beliefs. Efforts to address these challenges include education, awareness campaigns, and improved infrastructure.
Regulatory Compliance:

Blood banks adhere to strict regulatory standards to maintain the quality and safety of the blood supply. Compliance with regulations ensures the well-being of both donors and recipients.
Technological Advancements:

Advancements in technology, such as automation in blood testing and inventory management, contribute to increased efficiency and accuracy in blood bank operations.
Blood banks are integral to healthcare systems worldwide, serving as a lifeline for patients in need of blood transfusions for various medical conditions and emergencies.

CHAPTER-2

PROJECT ANALYSIS

2.1 EXISTING SYSTEM :

The existing system of a blood bank typically involves a combination of manual and technological processes to manage the collection, testing, storage, and distribution of blood and its components. Here are key aspects of the current blood bank system:

Donor Registration and Screening:

Donors are registered manually or through computer systems. Screening processes include interviews and medical history reviews to ensure donor eligibility.

Blood Collection:

Blood donation is usually done through manual phlebotomy techniques. The collected blood is then labeled with donor information for traceability.

Laboratory Testing:

Blood samples undergo various tests to determine blood type, screen for infectious diseases (such as HIV, hepatitis, and syphilis), and ensure compatibility with the intended recipient.

Blood Processing and Component Separation:

After testing, blood is processed to separate it into different components, including red blood cells, plasma, platelets, and other blood products.

Inventory Management:

Blood banks maintain manual or computerized inventory systems to track the quantity, type, and expiration dates of blood and its components. This helps prevent shortages and wastage.

Cross-Matching:

Before transfusions, blood samples from donors and recipients are cross-matched to

ensure compatibility and minimize the risk of transfusion reactions.

Distribution and Transfusion:

Blood products are distributed to hospitals and medical facilities based on their needs. Transfusions are administered under the supervision of healthcare professionals.

Documentation and Record Keeping:

Records of donor information, test results, and transfusions are maintained manually or through electronic databases. This documentation is crucial for traceability, audits, and regulatory compliance.

Emergency Response:

Blood banks play a vital role in responding to emergencies by providing a quick and efficient supply of blood products to hospitals treating trauma patients or during disaster situations.

Community Engagement and Awareness:

Blood banks often engage in community outreach programs to raise awareness about the importance of blood donation, encourage voluntary donors, and dispel myths surrounding blood donation.

Regulatory Compliance:

Blood banks adhere to local and international regulatory standards to ensure the safety, quality, and ethical handling of blood products.

Collaboration and Networking:

Blood banks may collaborate with other healthcare institutions, organizations, and government bodies to enhance their capabilities, share resources, and address challenges collectively.

While many blood banks have integrated technology into their processes, there is a continued effort to improve automation, enhance data management systems, and address challenges such as blood shortages and logistical issues

2.2 DRAW BACKS OF EXISTING SYSTEM:

1. Data Inaccuracy

2. Time Consuming

3. Limited Accessibility

4. Inventory Management Issues

5. Difficult Communication

6. Security Concerns

2.3 PROPOSED SYSTEM:

A comprehensive blood bank management system aims to streamline and optimize the entire process of blood donation, testing, storage, and distribution. Implement a user-friendly donor registration system, allowing donors to sign up online or through dedicated kiosks at blood banks.Maintain a centralized donor database with detailed information, including contact details, medical history, and donation history. Enable donors to schedule donation appointments online, reducing wait times and ensuring a steady supply of blood. Implement a robust inventory management system to track blood and blood component levels, expiration dates, and real-time updates on available stock

CHAPTER-3

REQUIREMENT ANALYSIS & SPECIFICATION

3.1 HARDWARE REQUIREMENT:

- Processor: Pentium
- RAM: 4GB
- Hard Disk: 1TB
- Speed: 1.1GHz

3.2 SOFTWARE REQUIREMENT:

- Operating System: Windows
- Scripting Language: JavaScript
- Back-End: Nodejs, Express,
- Front-End: React,css
- Supporting Tools: VSCode
- Database: MongoDB
- Type: Web Application
- Server: LiveServer(VSCode)

CHAPTER-4

TOOLS AND TECHNOLOGIES USED

4.1 INTRODUCTION TO VSCode

VSCode, short for Visual Studio Code, is a free, open-source code editor developed by Microsoft. It has gained immense popularity among developers for its lightweight design, extensive features, and support for various programming languages. Here are some key aspects of VSCode:

Cross-Platform:

VSCode is compatible with Windows, macOS, and Linux, making it a versatile choice for developers using different operating systems.

User-Friendly Interface:

The editor features a clean and intuitive interface with a sidebar for file navigation, a powerful integrated terminal, and a customizable layout to suit individual preferences.

Extensibility:

One of VSCode's strengths is its rich ecosystem of extensions. Developers can enhance and customize their coding experience by installing extensions for different languages, themes, and tools directly from the Visual Studio Code Marketplace.

Integrated Development Environment (IDE) Features:

Despite being a code editor, VSCode includes many features commonly associated with full-fledged integrated development environments, such as debugging, Git integration, IntelliSense (code completion and suggestions), and more.

Language Support:

VSCode supports a wide range of programming languages out of the box, including but not limited to JavaScript, TypeScript, Python, Java, C#, HTML, CSS, and more. Language-specific extensions further extend its capabilities.

Version Control:

Git integration is built into VSCode, allowing developers to manage version control directly within the editor. This includes features like committing changes, viewing commit history, and resolving merge conflicts.

Customizable and Configurable:

VSCode is highly customizable. Users can personalize the editor's appearance, configure key bindings, and modify settings to create a coding environment that suits their workflow.

Built-in Terminal:

VSCode includes an integrated terminal, allowing developers to run shell commands, scripts, or other tools without leaving the editor.

Live Share (Optional):

VSCode Live Share is an extension that enables real-time collaboration between developers. It allows multiple developers to work on the same codebase simultaneously.

Active Community and Support:

VSCode has a large and active community of developers. Regular updates, extensions, and support from both Microsoft and the community contribute to its ongoing improvement.

Whether you're working on web development, software engineering, or data science, Visual Studio Code provides a versatile and feature-rich environment for coding and collaboration.

4.2 INTRODUCTION TO CSS(Cascading Style)

CSS (Cascading Style Sheets) is used in React to style and visually enhance user interfaces. React applications are built with components, and each component may have its unique style requirements. CSS allows developers to apply styles to individual components or create global styles for the entire application, ensuring a consistent and appealing user experience.

React encourages a component-based architecture, where each component can encapsulate its structure, behavior, and style. CSS helps achieve this modularity by allowing developers to define styles that are specific to each component, making it easier to manage and maintain the codebase.

Additionally, CSS enables the creation of responsive and dynamic layouts, making React applications adaptable to various screen sizes and devices. With the rise of CSS-in-JS solutions, such as Styled Components or Emotion, developers can also integrate styles directly within their JavaScript code, enhancing the component-based nature of React.

In summary, CSS is a crucial tool in React development, providing a flexible and scalable approach to styling components, ensuring a visually cohesive and user-friendly application.

4.3 INTRODUCTION TO React

React is a powerful JavaScript library developed by Facebook for building modern, interactive user interfaces. Launched in 2013, React has since gained widespread popularity in web development due to its efficiency and declarative nature. It follows a component-based architecture, where the UI is broken down into modular and reusable components, promoting code reusability and maintainability.

React employs a virtual DOM (Document Object Model) to optimize rendering, allowing developers to create dynamic and responsive applications without compromising performance. Its unidirectional data flow ensures a predictable and easy-to-understand structure, enhancing the development process.

One of React's distinctive features is JSX (JavaScript XML), a syntax extension that enables the mixing of HTML-like code within JavaScript. This simplifies the creation of components and enhances code readability.

Furthermore, React has a vibrant ecosystem with tools like React Router for navigation, Redux for state management, and a thriving community contributing to a vast array of libraries and components. Its versatility extends beyond web applications to native mobile app development with React Native.

In summary, React has revolutionized front-end development, providing developers with a robust framework for building efficient, scalable, and maintainable user interfaces..

4.4 What is Nodejs and Express

Node.js and Express.js are a dynamic duo in modern web development, revolutionizing the way server-side applications are built. Node.js is a runtime environment that executes JavaScript code server-side, allowing developers to use JavaScript for both client and server applications. It utilizes the V8 JavaScript engine, originally developed by Google for Chrome.

Express.js, commonly referred to as Express, is a minimalist and flexible web application framework for Node.js. It provides a set of robust features and tools to build web and mobile applications, making the process more efficient and streamlined. Express simplifies tasks such as routing, handling HTTP requests and responses, and managing middleware, enabling developers to focus on building functionality rather than dealing with boilerplate code.

Together, Node.js and Express empower developers to create scalable, high-performance web applications. Node.js's non-blocking, event-driven architecture ensures asynchronous processing, making it well-suited for handling numerous concurrent connections. Express enhances this by providing a structured and organized framework that facilitates the creation of RESTful APIs and web servers.

This dynamic duo has become a popular choice for developers seeking a JavaScript-centric, efficient, and scalable solution for server-side development. Whether building APIs, microservices, or full-stack applications, Node.js and Express.js provide a robust foundation for creating modern, responsive, and high-performance web applications.

4.5 What is SERVER

A server is a computer or system that provides resources, data, services, or programs to other computers, known as clients, over a network. In theory, whenever computers share resources with client machines they are considered servers.

The term "server" can refer to both hardware and software components, but let's focus on the software aspect, specifically in the context of web servers.
A web server is a software application or program responsible for handling client requests and serving web content over the internet. Here's a simplified overview of how a web server works:
Listening for Requests:
A web server listens for incoming requests from clients (typically web browsers) on a specified port (commonly port 80 for HTTP and port 443 for HTTPS).
Receiving Requests:
When a user makes a request by entering a URL in a web browser or clicking on a link, the browser sends an HTTP request to the server, specifying the desired resource (e.g., a web page or an image).

Processing Requests:
The web server processes the incoming request by determining the requested resource and identifying the appropriate action to take. This may involve retrieving data from a database, running server-side scripts, or accessing static files.
Generating Responses:
Based on the request, the web server generates an appropriate response. This response includes the requested content, along with metadata such as HTTP headers and status codes.
Sending Responses:
The web server sends the response back to the client over the internet. The response contains the requested content, which could be HTML, CSS, JavaScript, images, or other types of files.
Client Rendering:
The client's web browser receives the response and renders the content, displaying it to the user. If the response includes additional resources (e.g., images, stylesheets), the browser may issue additional requests to the server to retrieve those resources.
Connection Handling:
The web server manages multiple connections simultaneously, handling requests from different clients concurrently. It employs techniques such as connection pooling to optimize resource usage and ensure responsiveness.
Popular web servers include Apache, Nginx, and Microsoft Internet Information Services (IIS). These servers play a crucial role in serving web content, supporting the functionality of websites, web applications, and various online services

4.6. What is MongoDb

MongoDB is a NoSQL database that uses a flexible, document-oriented model for data storage. It stores data in BSON format, a binary representation of JSON-like documents. This schema-less approach allows for dynamic and scalable data structures. MongoDB supports automatic sharding for horizontal scalability and provides high availability through replica sets. Developers appreciate its ease of use, powerful querying capabilities, and support for indexing, making it a popular choice for applications handling large volumes of diverse data.

Chapter-5

MODULES OF PROPOSED SYSTEM

This proposed system consists of 5 main modules, which are listed below.

5.1 INVENTORY MODULE

In a Blood Bank Management System, the Inventory module plays a critical role in efficiently managing and tracking blood and blood-related products. Its functions include:

Blood Product Management:Tracking the quantity and details of different blood products, including whole blood, red blood cells, plasma, and platelets.

Donor Information:Recording and managing information about blood donors, including their eligibility, contact details,and donation history.

Inventory Levels:Maintaining optimal inventory levels to meet the demands of healthcare facilities while minimizing waste or shortages.

Blood Typing and Cross-Matching:Recording blood types and facilitating cross-matching to ensure compatibility between donor blood and recipient blood, minimizing the risk of transfusion reactions.

Donation Campaigns:Managing information related to blood donation campaigns, including locations, dates, and results, to encourage community participation.

Testing and Screening:Integrating with the testing and screening processes to ensure that donated blood meets safety standards and regulatory requirements.

Reservations and Allocations:Reserving specific blood units for scheduled surgeries or medical procedures and allocating blood products based on medical requirements.

Transfusion History:Maintaining a comprehensive record of blood transfusions, including recipient details, transfusion dates, and the specific blood products used.

5.2 ORGANISATION MODULE

Here in which we have to see all the blood records in which type of blood , where is it in and out , what is quantity of blood, who has done in and out here , then see time and date it will show in this way. Here to see all donors in donar, how many doner here   Send application to him, whatever organisation this is . then here to see whichever hospital Will be registered on it **.**

5.3 DONAR MODULE

here to see which organisation to donate blood see here and it will show where and how much blood you have donate.

5.4 HOSPITAL MODULE

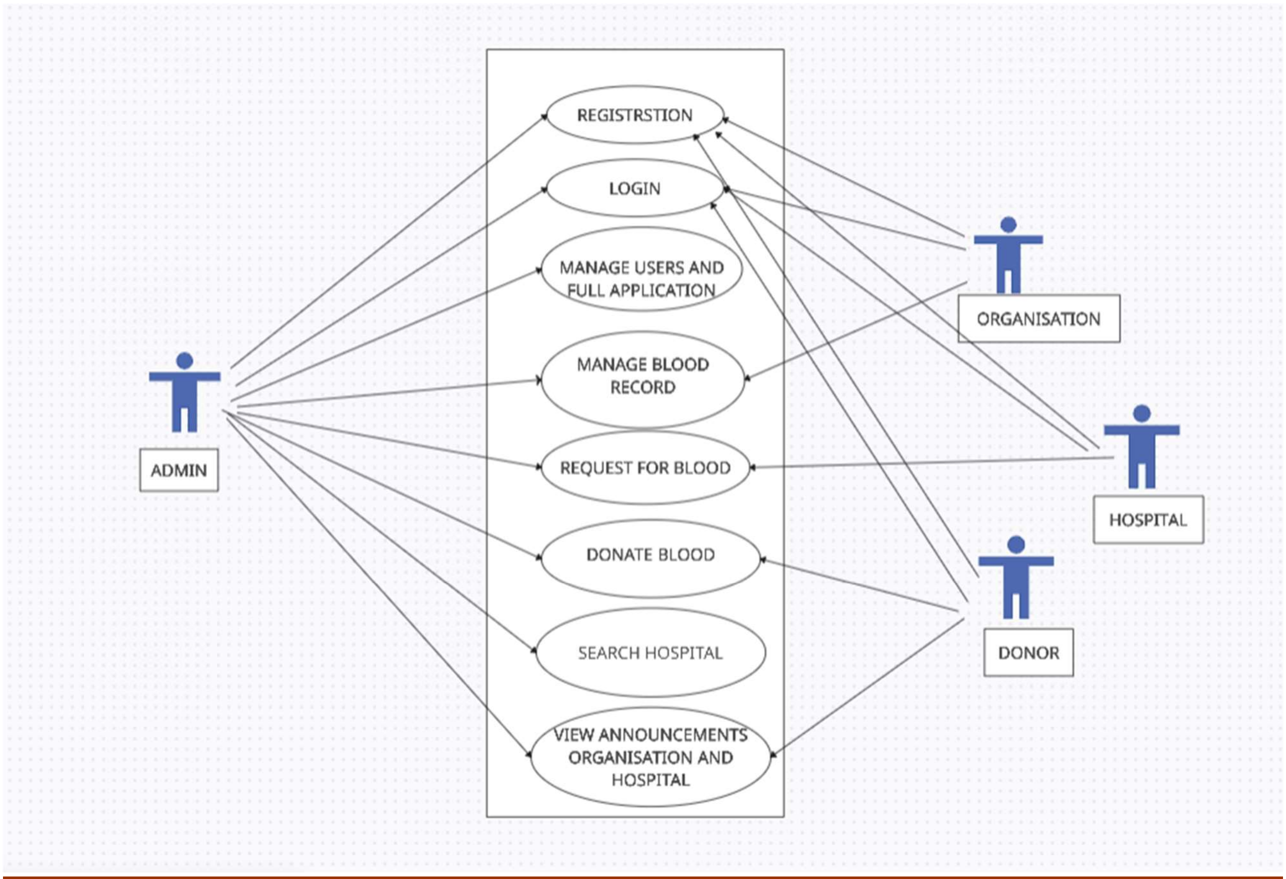
Here show organisation name and we can see the consumer and who has consumed our blood.

5.5 ADMIN MODULE

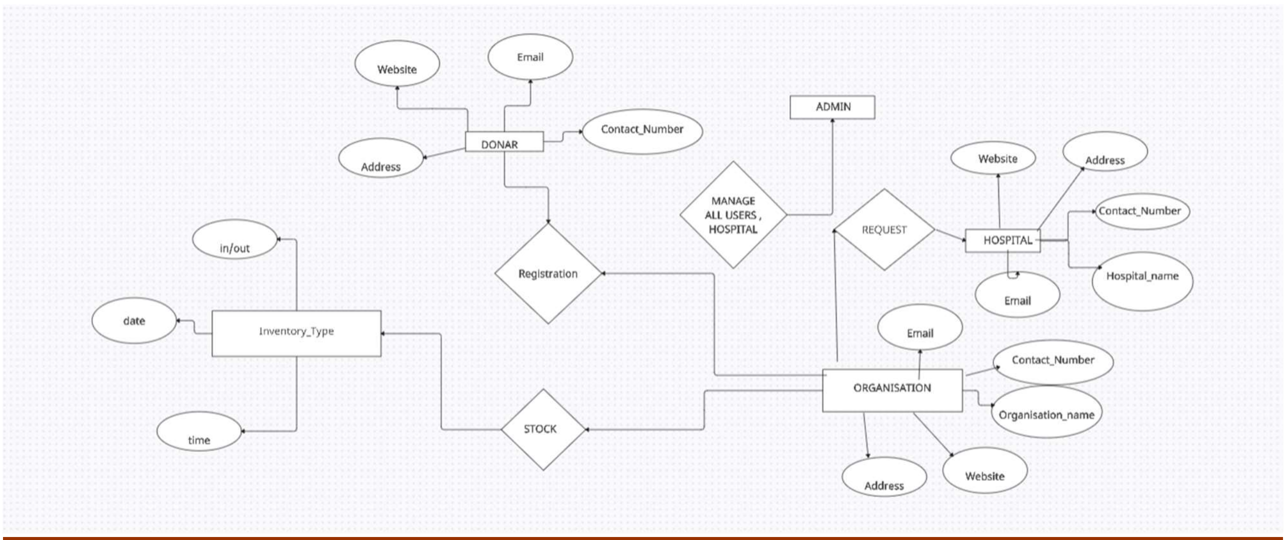
Admin manage the hospital, Doner, Organisation.Admin can delete the register of Hospital,Organisation and User.

SYSTEM DESIGN

Use-Case Diagram



E-RDiagram



TESTING AND RESULT

8.1 **Types Of Testing**

8.1.1 System Testing:

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and coding. The user tests the developed system and changes are made according to their needs. The testing phase involves the testing developed system using various kinds of data.

System is the stage of implementation that is aimed at assuring at the system works accurately and efficiently before live operation commences. Testing is vital to the success of the system. System testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved. The candidate system is subject to a variety of tests such as recover, security and usability tests. A series of testing is performed for the proposed system before the system is ready for the user acceptance testing.

Implementation ends with formal tests. The test data are very crucial to this process. They must be realistic and cover extreme conditions are well. Ideally, vary alternative path through the program should be exercised at least once beyond the test data. The system test must involve all the elements that compose the system including program validation checking, files, and forms and triggers procedures.

8.1.2 Component testing

- Testing of individual program components i.e. the each module is tested
- Usually the responsibility of the component developer (except sometimes for critical systems);
- Tests are derived from the developer's experience.

- Component or unit testing is the process of testing individual components in isolation.
- It is a defect testing process.
- Components may be:
- Individual functions or methods within an object;
- Object classes with several attributes and methods; Composite components with defined interfaces used to access their functionality

8.2 Testing Strategies

Following are few of the testing strategies used for the testing purpose:

- Unit testing.
- Validation testing.
- Output testing.
- User acceptance testing.

8.2.1 Unit Testing

Unit testing focuses effort on the smallest unit of software design of the module. This is also known as 'Module Testing'. The module of FSA system is tested separately. This testing was carried out during programming stage itself in this testing each module is found to be working satisfactorily with regards to the expected output from the module.

8.2.2 Validating Testing

At the culmination of integration testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected and final series of software test begins. Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software function in a manner that can be reasonably expected by the customer.

After validation test has been conducted, one of the two possible conditions exists, the functions are performance characteristics confirm to specification and are accepted or a deviation from specification is uncovered and deficiency list is created. Proposed system under consideration has been tested using validation testing and found to be working satisfactorily.

8.2.3 Output Testing

After performing the validation testing the next test is output testing of the proposed system since no system could be useful if it does not produce the required output in the specified format. Asking the user about the format required by them tests the outputs generated or displayed by the system under consideration. Here, the output format is considered in two ways. One on-screen and other is printed format. The output format on the screen is found to be correct as the format was designed in the system phase according to the user’s needs. Hence, output testing does not result in any correction in the system.

8.2.4 User Acceptance Testing

User acceptance of a system is the key factory for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the perspective system. Users at time of developing can make changes wherever required.

This is done in regards to the following points:

- Input screen design.
- Output Screen design.
- Menu driven system.
- Format of reports and other outputs

Taking various kinds of test data does the above tests. Preparation of the test data places a vital role in system testing. After preparing the test data the system under study is tested using the same. While testing the system by using the test, errors are uncovered. They are then corrected and noted down for future use.

8.3 Testing Guidelines

Testing guidelines are hints for the testing team to help them choose tests that will reveal defects in the system.

- Choose inputs that force the system to generate all error messages;
- Design inputs that cause buffers to overflow;
- Repeat the same input or input series several times;
- Force invalid outputs to be generated;
- Force computation results to be too large or too small.

8.4 Test Case Design

- Involves designing the test cases (inputs and outputs) used to test the system.
- The goal of test case design is to create a set of tests that are effective in validation and defect testing.
- Design approaches:

- Requirements-based testing;
- Partition testing;
- Structural testing.

8.4.1 Requirements based testing

- A general principle of requirements engineering is that requirements should be testable.
- Requirements-based testing is a validation testing technique where you consider each requirement and derive a set of tests for that requirement.

8.4.2 Partition testing

- Input data and output results often fall into different classes where all members of a class are related.
- Each of these classes is an equivalence partition or domain where the program behaves in an equivalent way for each class member.
- Test cases should be chosen from each partition.

8.4.3 Structural testing

- Sometime called white-box testing.
- Derivation of test cases according to program structure. Knowledge of the program is used to identify additional test cases.
- Objective is to exercise all program statements (not all path combination)

SCOPE

The scope of Blood Bank Management is broad and essential, encompassing various aspects of handling, managing, and optimizing blood-related processes. Here are key areas within the scope of Blood Bank Management:

Donor Management:Registration and maintenance of donor information, including eligibility criteria, contact details, and donation history.

Inventory Management:Tracking and managing blood and blood product inventory, ensuring optimal levels, preventing shortages, and minimizing wastage.

Blood Typing and Cross-Matching:Performing and managing blood typing tests and cross-matching to ensure compatibility between donors and recipients.

Testing and Screening:Implementing and overseeing testing procedures to screen donated blood for infectious diseases and ensuring the safety of blood products.

Transfusion Management:Efficiently managing the allocation and distribution of blood products to healthcare facilities based on demand and emergency situations.

Medical Records and Documentation:Maintaining comprehensive records of donor and recipient information, transfusion history, and medical testing results for compliance and traceability.

Donation Campaigns and Community Engagement:Organizing and managing blood donation campaigns to encourage community participation, raise awareness, and maintain a sustainable donor pool.

Reporting and Analytics:Generating reports and analytics on blood inventory levels, donation patterns, and other key performance indicators to support decision-making.

Staff Management:Managing staff information, roles, and responsibilities within the blood bank, ensuring efficient operation and coordination.

Regulatory Compliance:Adhering to local and international regulatory standards and guidelines to ensure the safety, quality, and ethical handling of blood products.

Integration with Healthcare Systems:Integrating with hospital or healthcare information systems to facilitate seamless communication and coordinated patient care.

Emergency Response and Disaster Preparedness:Developing protocols for responding to emergencies, disasters, and unexpected surges in demand for blood products.

Technological Integration:Implementing technology solutions, such as information systems, barcoding, and RFID technology, to enhance efficiency, accuracy, and traceability.

CODE

Frontend

APP.JS

```
import HomePage from "./pages/HomePage";
import Login from "./pages/auth/Login";
import Register from "./pages/auth/Register";
import { ToastContainer } from "react-toastify";
import "react-toastify/dist/ReactToastify.css";
import ProtectedRoute from "./components/Routes/ProtectedRoute";
import PublicRoute from "./components/Routes/PublicRoute";
import Donar from "./pages/Dashboard/Donar";
import Hospitals from "./pages/Dashboard/Hospitals";
import OrganisationPage from "./pages/Dashboard/OrganisationPage";
import Consumer from "./pages/Dashboard/Consumer";
import Donation from "./pages/Donation";
import Analytics from "./pages/Dashboard/Analytics";
import DonarList from "./pages/Admin/DonarList";
import HospitalList from "./pages/Admin/HospitalList";
import OrgList from "./pages/Admin/OrgList";
import AdminHome from "./pages/Admin/AdminHome";
function App() {
  return (
    <>
      <ToastContainer />
      <Routes>
        <Route
      path="/admin"
      element={
        <ProtectedRoute>
          <AdminHome />
        </ProtectedRoute>
      }
    />
    <Route
      path="/donar-list"
      element={
        <ProtectedRoute>
          <DonarList />
        </ProtectedRoute>
      }
    />
    <Route
      path="/org-list"
      element={
        <ProtectedRoute>
          <OrgList />
        </ProtectedRoute>
```

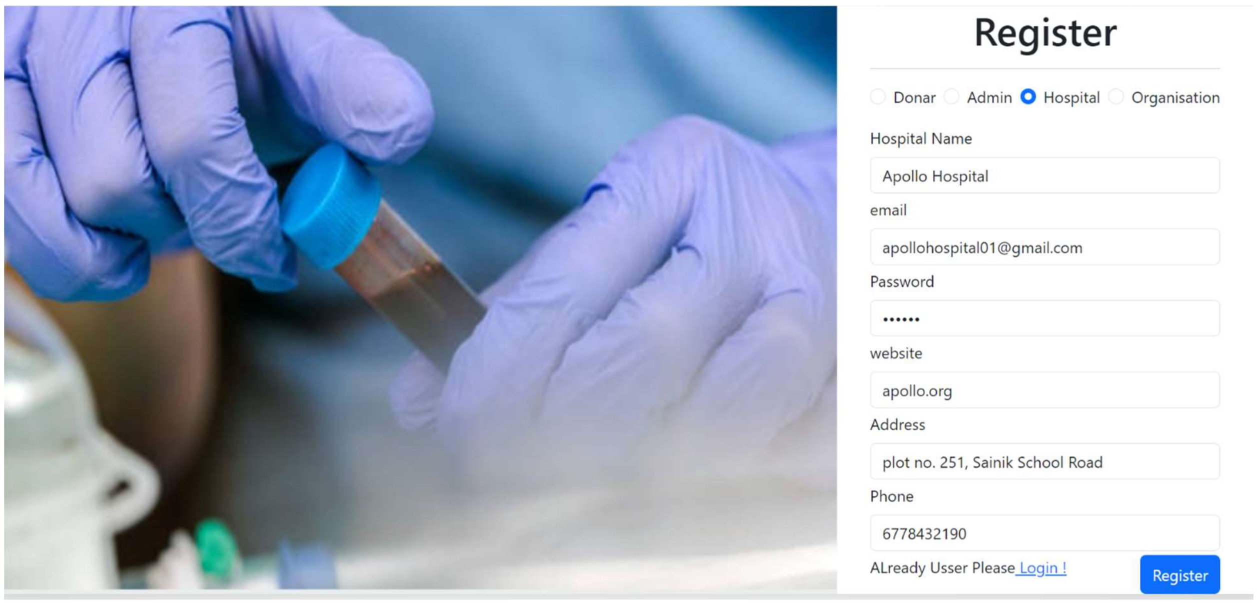
Backend

SERVER.JS

```
const express = require("express");
const dotenv = require("dotenv");
const colors = require("colors");
const morgan = require("morgan");
const cors = require("cors");
const connectMongoDb = require("./config/db");
//dot config
dotenv.config();

//mongodb connection
  console.log("Successfull Connected with mongoDb");
});

//rest object (Store thev Express features)
const app = express();

//middlewares
app.use(express.json());
app.use(cors());
app.use(morgan("dev"));

//routes
app.use("/api/v1/auth", require("./routes/authRoutes"));
app.use("/api/v1/inventory", require("./routes/inventoryRoutes"));
app.use("/api/v1/analytics", require("./routes/analyticsRoutes"));
app.use("/api/v1/admin", require("./routes/adminRoutes"));
```

OUTPUT SCREEN

REGISTRATION PAGE



Register

Donar Admin [✓ Hospital] Organisation

Hospital Name

Apollo Hospital

email

apollohospital01@gmail.com

Password

••••••

website

apollo.org

Address

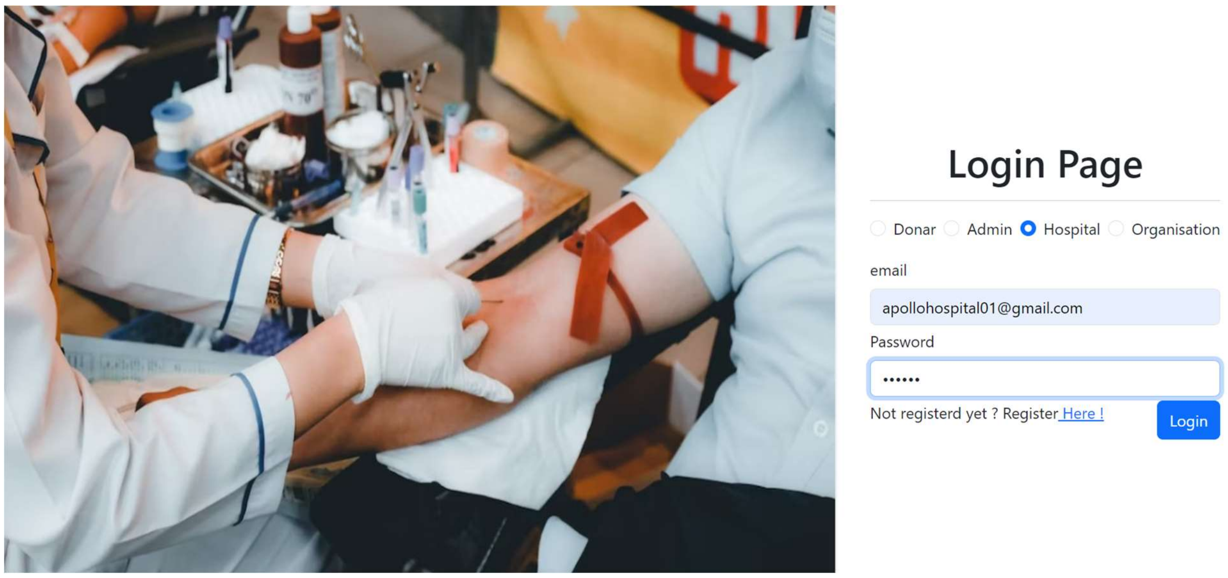
plot no. 251, Sainik School Road

Phone

6778432190

ALready Usser Please Login !

LOGIN PAGE



Login Page

Donar Admin [✓ Hospital] Organisation

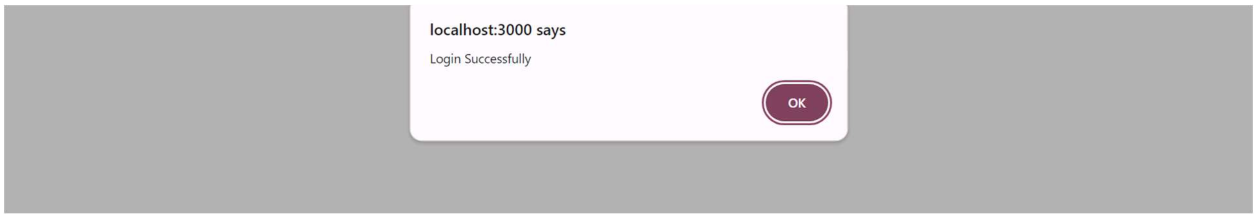
email

apollohospital01@gmail.com

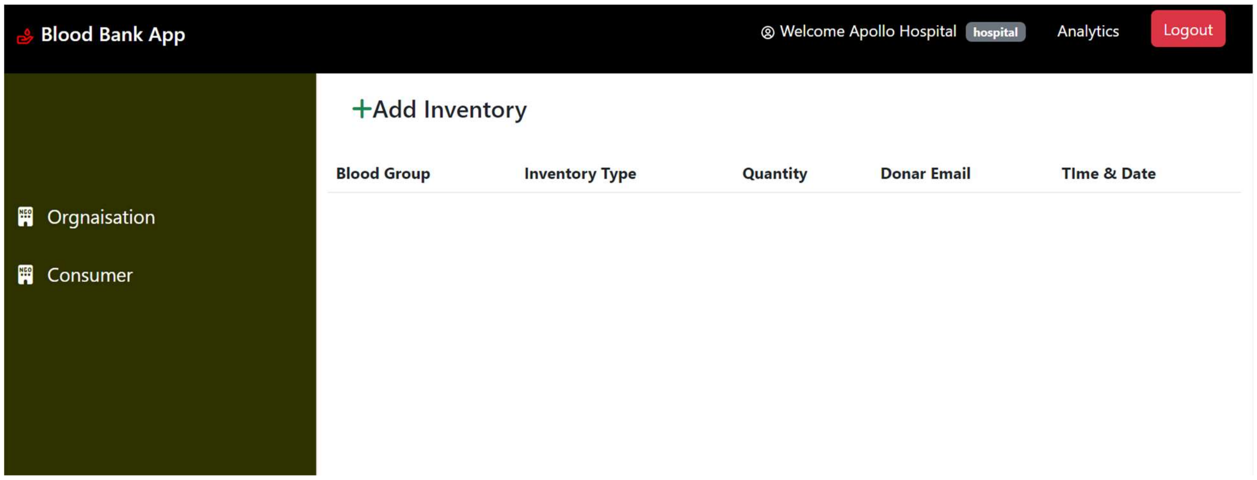
Password

••••••

Not registerd yet ? Register Here !

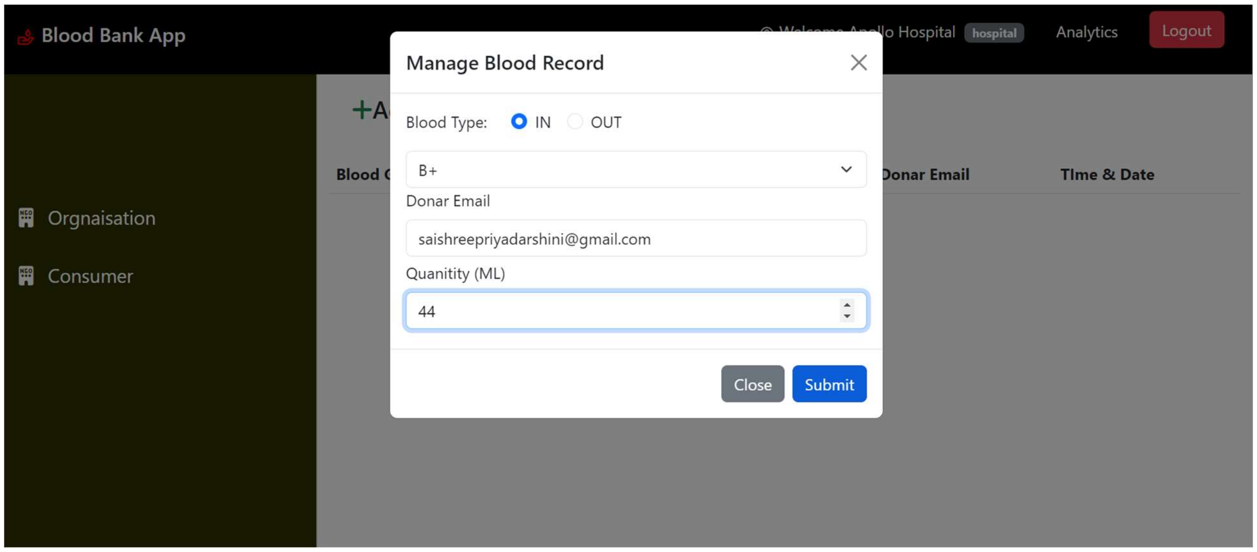

localhost:3000 says

Login Successfully



Blood Group	Inventory Type	Quantity	Donar Email	TIme & Date

ADD INVENTORY


Manage Blood Record

Blood Type: [✓ IN] OUT

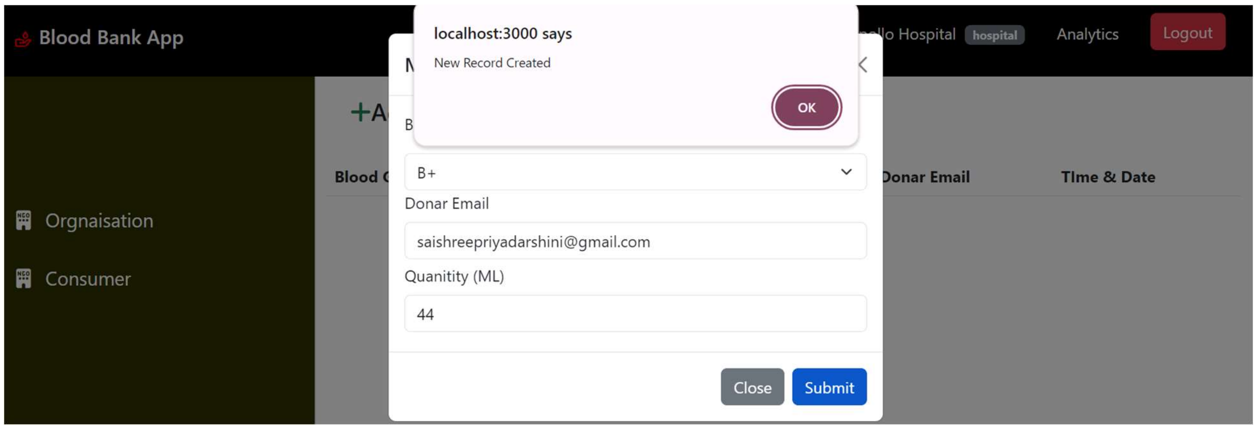
B+

Donar Email

saishreepriyadarshini@gmail.com

Quanitity (ML)

44



Blood Bank App

localhost:3000 says

New Record Created

OK

B+

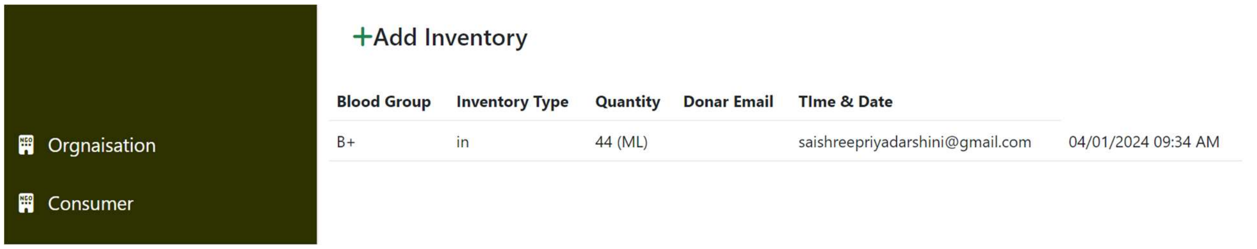
Donar Email

saishreepriyadarshini@gmail.com

Quanitity (ML)

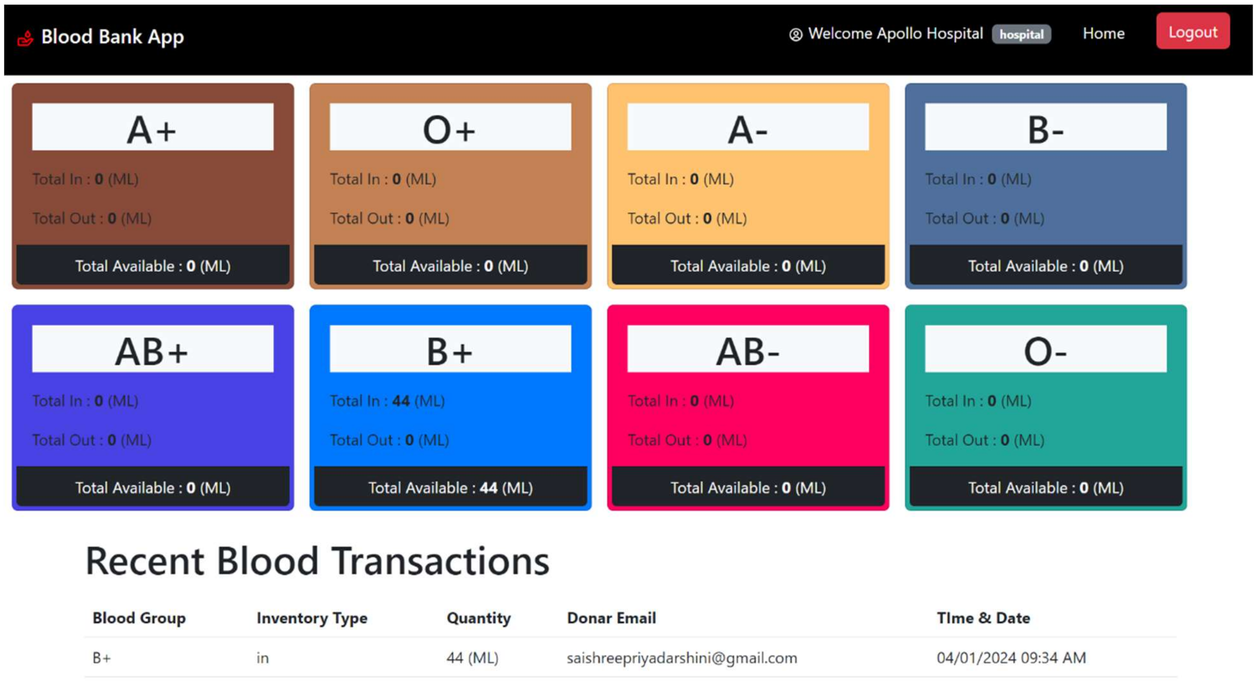
44

Close Submit



+Add Inventory

Blood Group	Inventory Type	Quantity	Donar Email	TIme & Date
B+	in	44 (ML)	saishreepriyadarshini@gmail.com	04/01/2024 09:34 AM



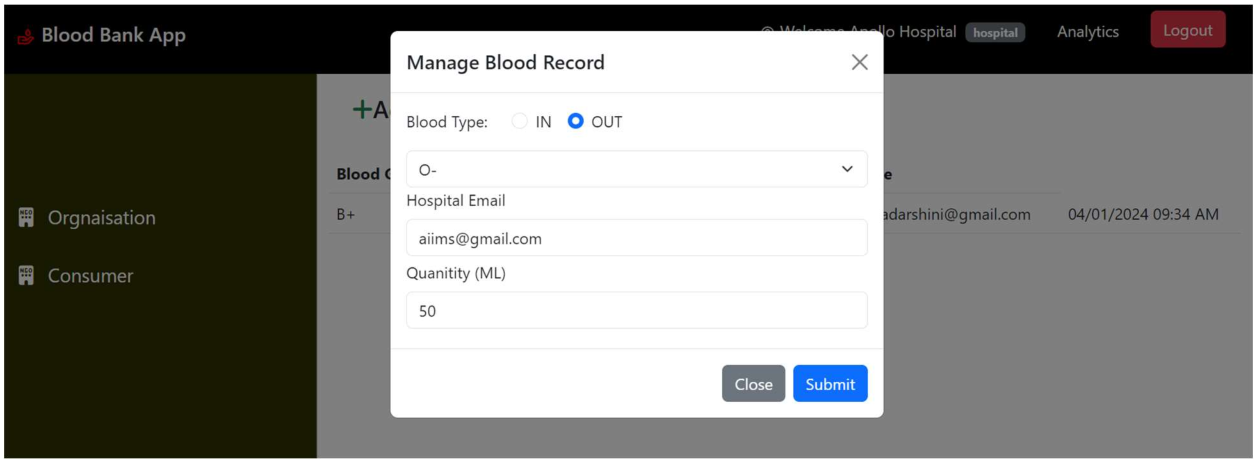
Blood Bank App

Welcome Apollo Hospital hospital

A+ Total In : **0** (ML) Total Out : **0** (ML) Total Available : **0** (ML)	O+ Total In : **0** (ML) Total Out : **0** (ML) Total Available : **0** (ML)	A- Total In : **0** (ML) Total Out : **0** (ML) Total Available : **0** (ML)	B- Total In : **0** (ML) Total Out : **0** (ML) Total Available : **0** (ML)
AB+ Total In : **0** (ML) Total Out : **0** (ML) Total Available : **0** (ML)	B+ Total In : **44** (ML) Total Out : **0** (ML) Total Available : **44** (ML)	AB- Total In : **0** (ML) Total Out : **0** (ML) Total Available : **0** (ML)	O- Total In : **0** (ML) Total Out : **0** (ML) Total Available : **0** (ML)

Recent Blood Transactions

Blood Group	Inventory Type	Quantity	Donar Email	TIme & Date
B+	in	44 (ML)	saishreepriyadarshini@gmail.com	04/01/2024 09:34 AM



Manage Blood Record

Blood Type: IN [✓ OUT]

O-

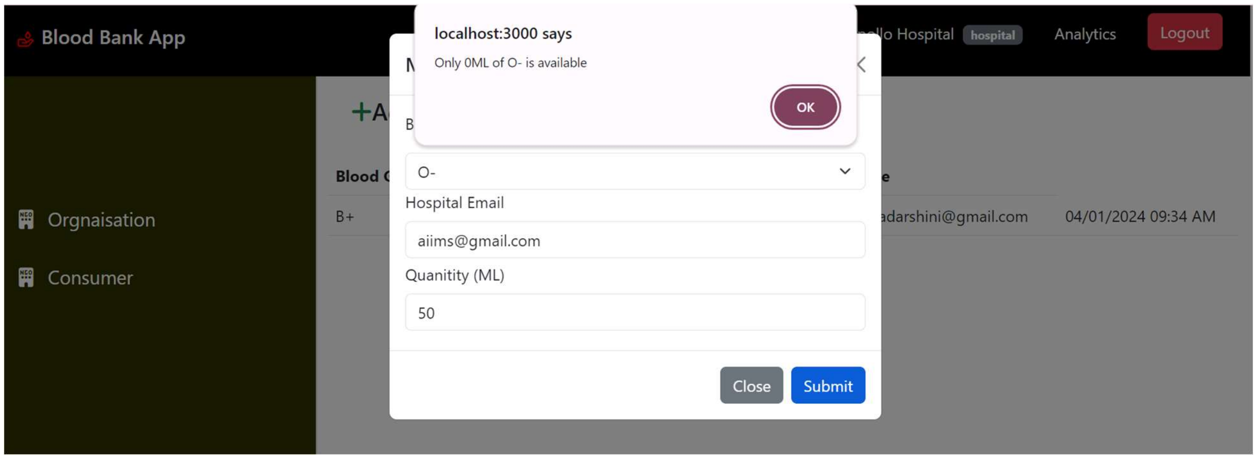
Hospital Email

aiims@gmail.com

Quanitity (ML)

50

B+	adarshini@gmail.com	04/01/2024 09:34 AM



localhost:3000 says

Only 0ML of O- is available

O-

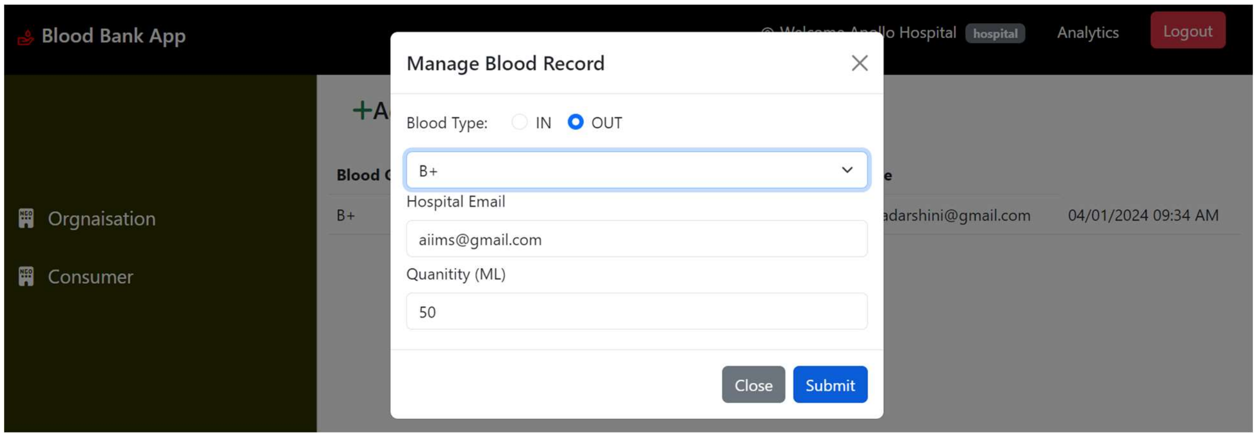
Hospital Email

aiims@gmail.com

Quanitity (ML)

50

B+	adarshini@gmail.com	04/01/2024 09:34 AM



Manage Blood Record

Blood Type: IN [✓ OUT]

B+

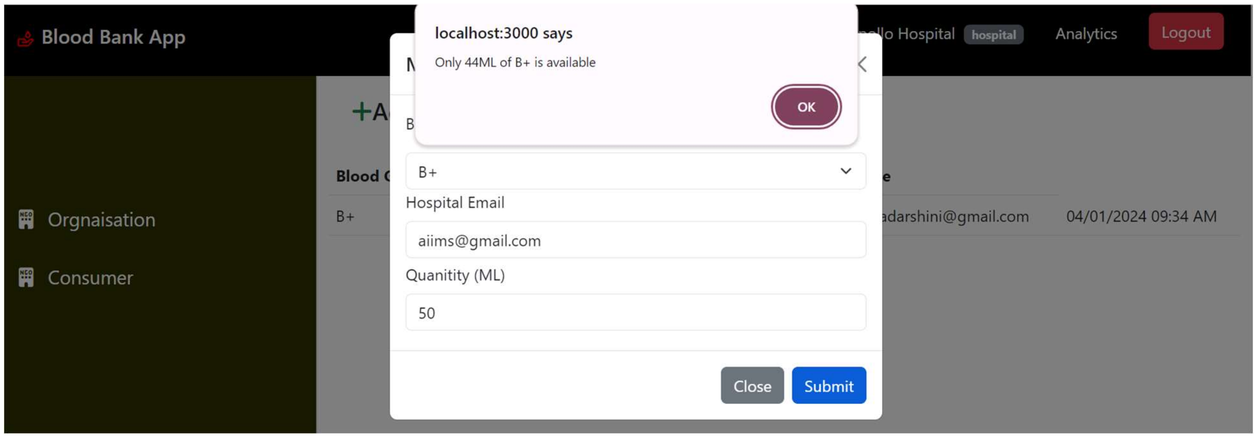
Hospital Email

aiims@gmail.com

Quanitity (ML)

50

B+	adarshini@gmail.com	04/01/2024 09:34 AM


Blood Bank App

Orgnaisation

Consumer

localhost:3000 says

Only 44ML of B+ is available

OK

B+

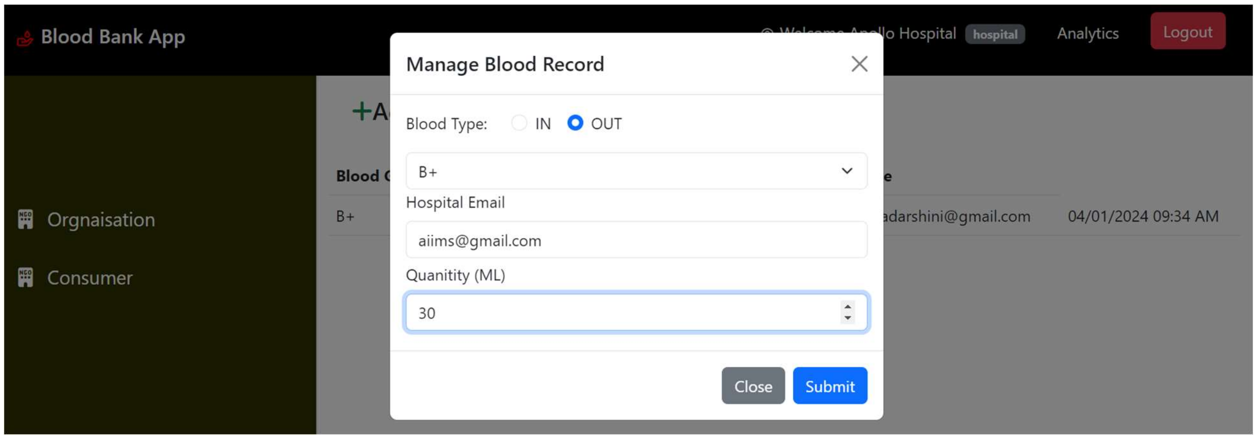
Hospital Email

aiims@gmail.com

Quanitity (ML)

50

Close Submit


Blood Bank App

Orgnaisation

Consumer

Manage Blood Record

Blood Type: IN [✓ OUT]

B+

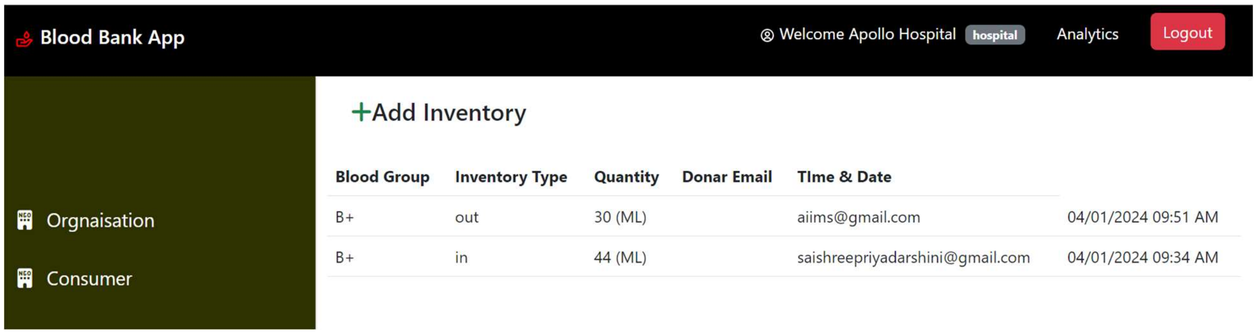
Hospital Email

aiims@gmail.com

Quanitity (ML)

30

Close Submit


Blood Bank App

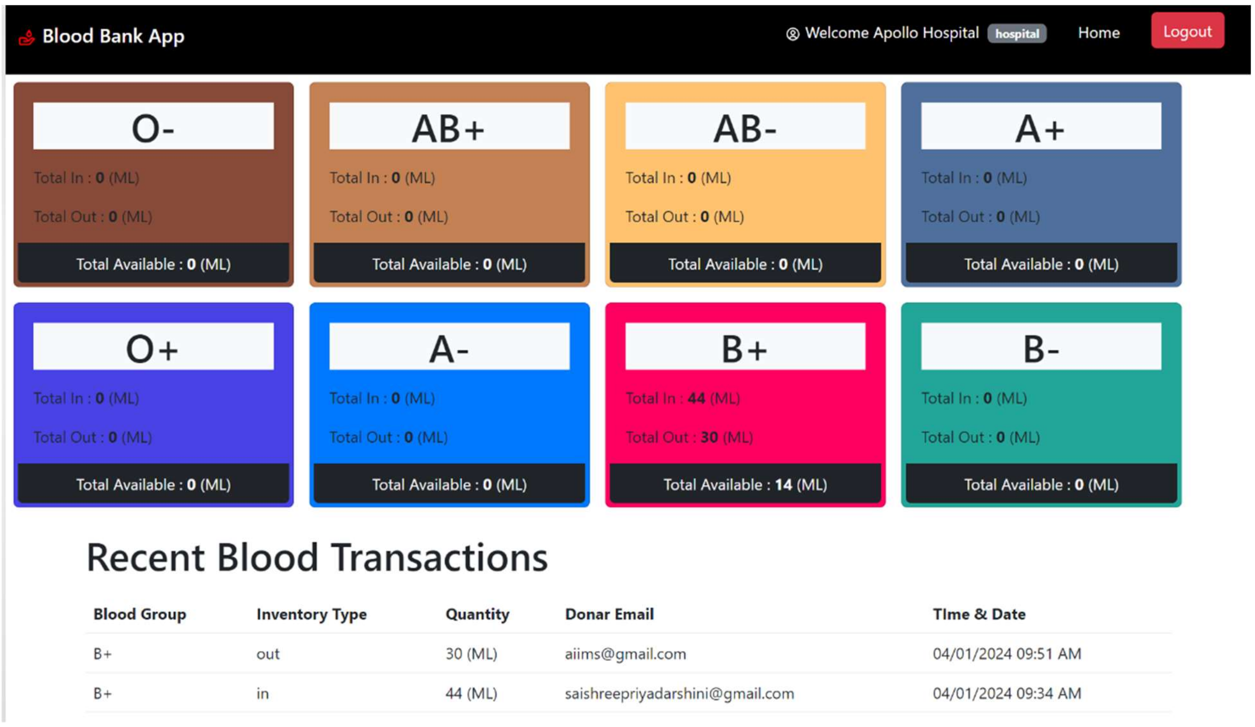
Welcome Apollo Hospital hospital Analytics Logout

Orgnaisation

Consumer

+Add Inventory

Blood Group	Inventory Type	Quantity	Donar Email	TIme & Date
B+	out	30 (ML)	aiims@gmail.com	04/01/2024 09:51 AM
B+	in	44 (ML)	saishreepriyadarshini@gmail.com	04/01/2024 09:34 AM

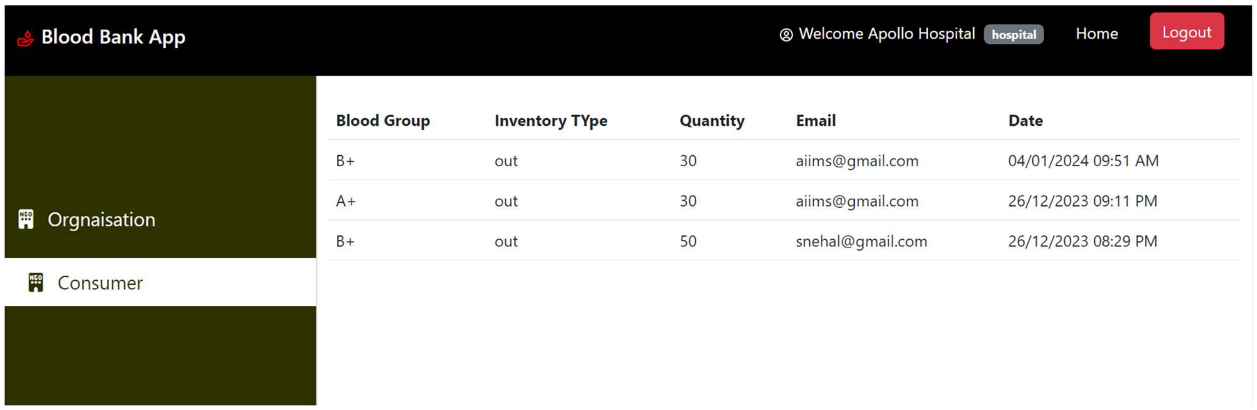


Blood Bank App | Welcome Apollo Hospital hospital | Home | Logout

Blood Group	Total In	Total Out	Total Available
O-	Total In : **0** (ML)	Total Out : **0** (ML)	Total Available : **0** (ML)
AB+	Total In : **0** (ML)	Total Out : **0** (ML)	Total Available : **0** (ML)
AB-	Total In : **0** (ML)	Total Out : **0** (ML)	Total Available : **0** (ML)
A+	Total In : **0** (ML)	Total Out : **0** (ML)	Total Available : **0** (ML)
O+	Total In : **0** (ML)	Total Out : **0** (ML)	Total Available : **0** (ML)
A-	Total In : **0** (ML)	Total Out : **0** (ML)	Total Available : **0** (ML)
B+	Total In : **44** (ML)	Total Out : **30** (ML)	Total Available : **14** (ML)
B-	Total In : **0** (ML)	Total Out : **0** (ML)	Total Available : **0** (ML)

Recent Blood Transactions

Blood Group	Inventory Type	Quantity	Donar Email	TIme & Date
B+	out	30 (ML)	aiims@gmail.com	04/01/2024 09:51 AM
B+	in	44 (ML)	saishreepriyadarshini@gmail.com	04/01/2024 09:34 AM

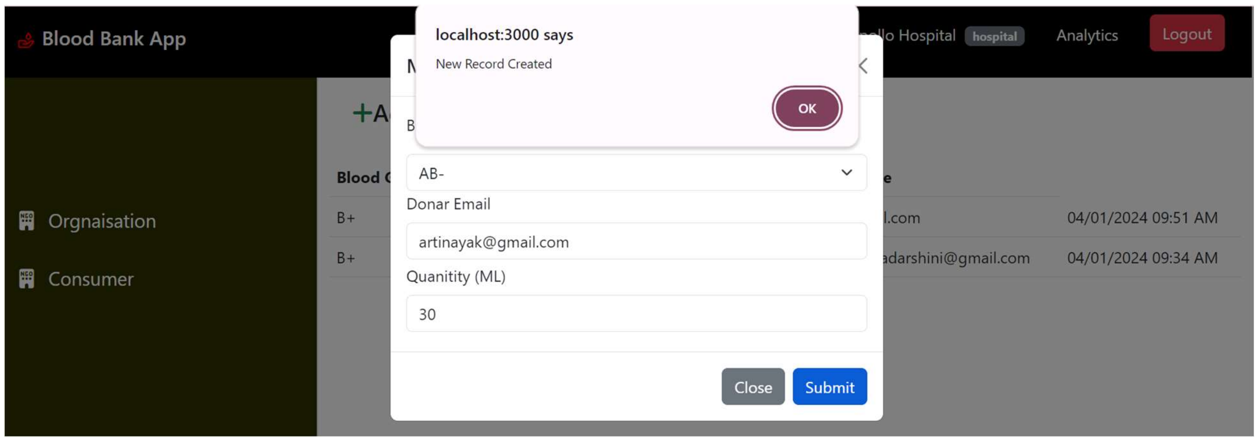


Blood Bank App | Welcome Apollo Hospital hospital | Home | Logout

Orgnaisation

Consumer

Blood Group	Inventory TYpe	Quantity	Email	Date
B+	out	30	aiims@gmail.com	04/01/2024 09:51 AM
A+	out	30	aiims@gmail.com	26/12/2023 09:11 PM
B+	out	50	snehal@gmail.com	26/12/2023 08:29 PM



Blood Bank App | llo Hospital hospital | Analytics | Logout

Orgnaisation

Consumer

localhost:3000 says

New Record Created

OK

AB-

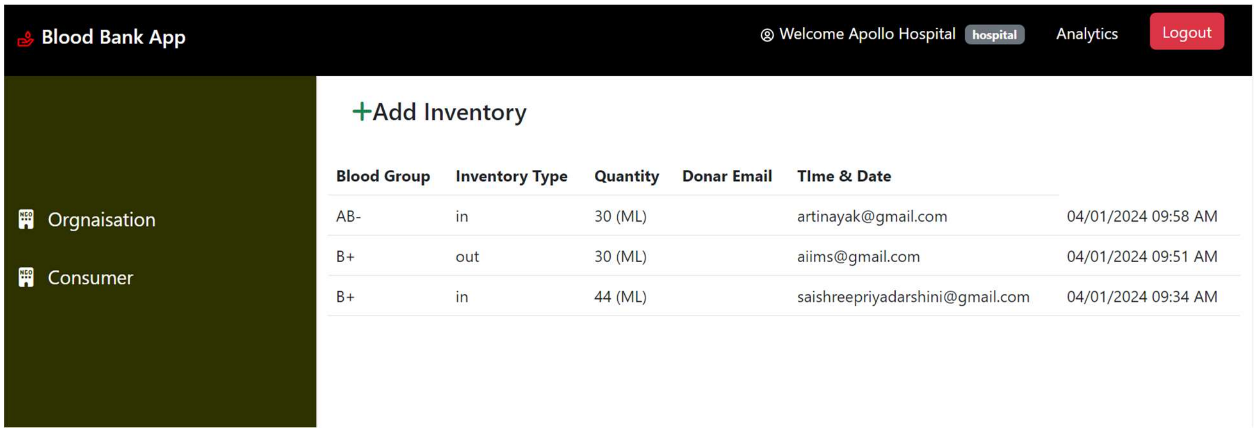
Donar Email

artinayak@gmail.com

Quanitity (ML)

30

Close | Submit


Blood Bank App

Welcome Apollo Hospital hospital

Analytics

Logout

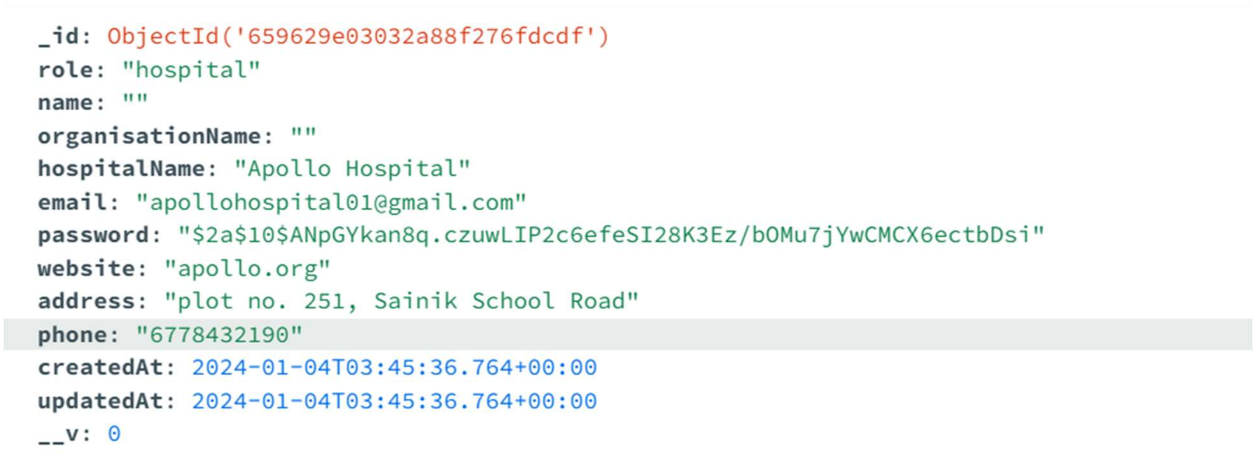
Orgnaisation

Consumer

+Add Inventory

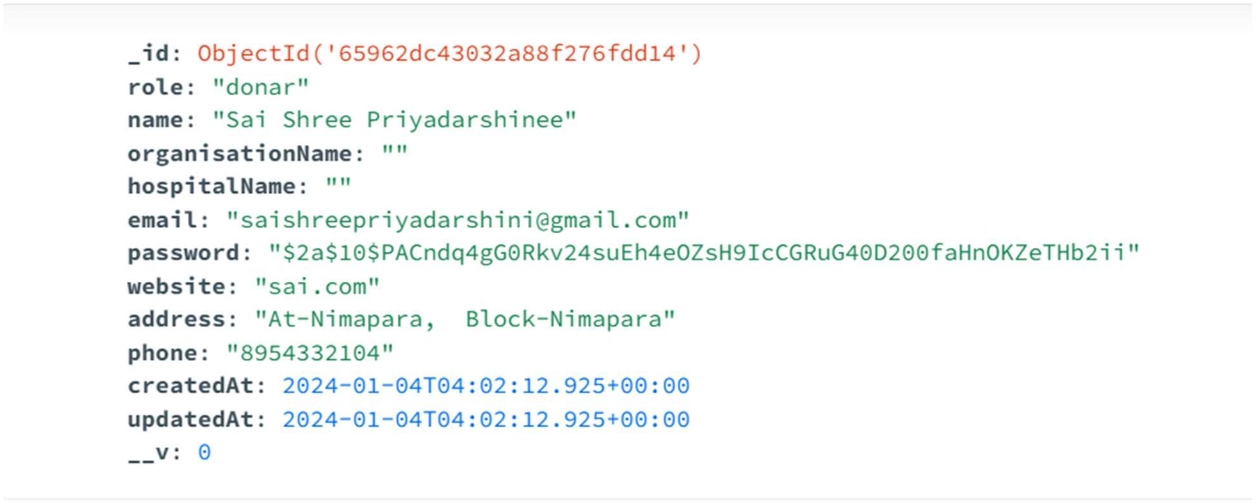
Blood Group	Inventory Type	Quantity	Donar Email	TIme & Date
AB-	in	30 (ML)	artinayak@gmail.com	04/01/2024 09:58 AM
B+	out	30 (ML)	aiims@gmail.com	04/01/2024 09:51 AM
B+	in	44 (ML)	saishreepriyadarshini@gmail.com	04/01/2024 09:34 AM

HOSPITAL DATABASE



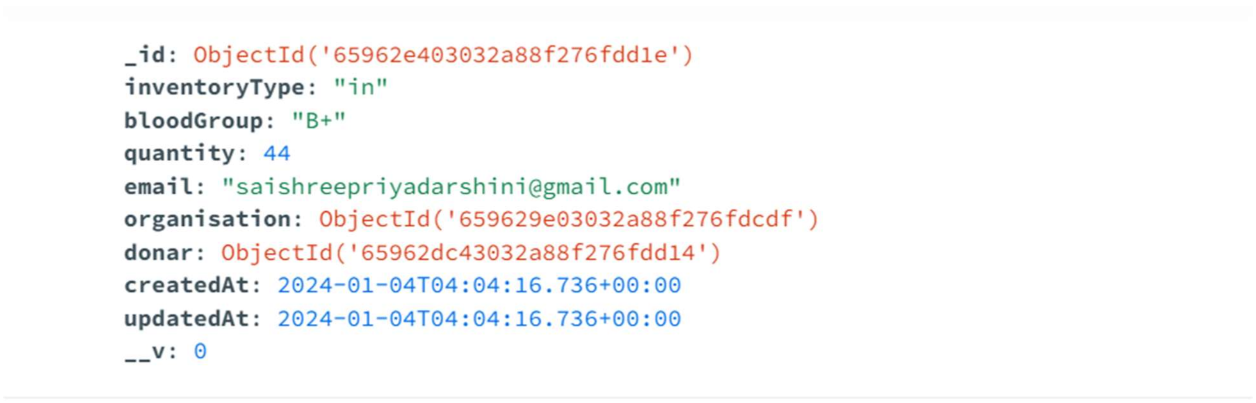
```
_id: ObjectId('659629e03032a88f276fdcdf')
role: "hospital"
name: ""
organisationName: ""
hospitalName: "Apollo Hospital"
email: "apollohospital01@gmail.com"
password: "$2a$10$ANpGYkan8q.czuwLIP2c6efeSI28K3Ez/bOMu7jYwCMCX6ectbDsi"
website: "apollo.org"
address: "plot no. 251, Sainik School Road"
phone: "6778432190"
createdAt: 2024-01-04T03:45:36.764+00:00
updatedAt: 2024-01-04T03:45:36.764+00:00
__v: 0
```

DONAR DATABASE

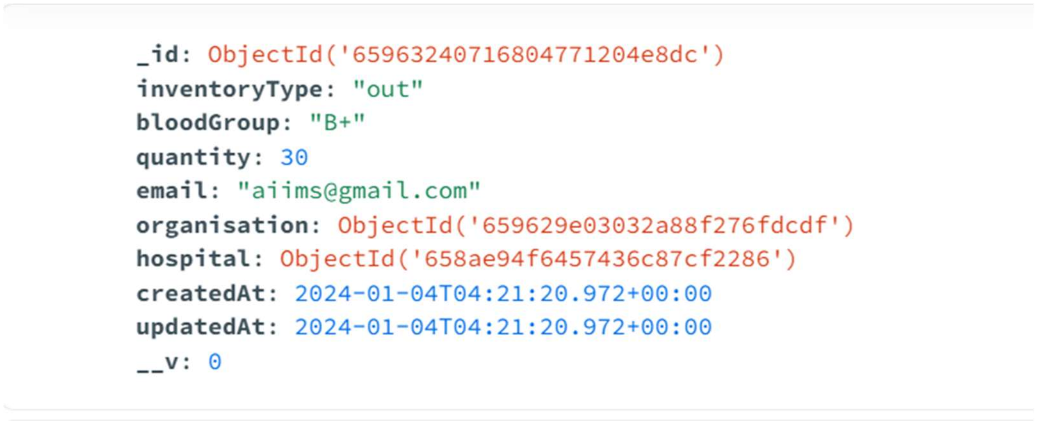


```
_id: ObjectId('65962dc43032a88f276fdd14')
role: "donar"
name: "Sai Shree Priyadarshinee"
organisationName: ""
hospitalName: ""
email: "saishreepriyadarshini@gmail.com"
password: "$2a$10$PACndq4gG0Rkv24suEh4eOZsH9IcCGRuG40D200faHnOKZeTHb2ii"
website: "sai.com"
address: "At-Nimapara,  Block-Nimapara"
phone: "8954332104"
createdAt: 2024-01-04T04:02:12.925+00:00
updatedAt: 2024-01-04T04:02:12.925+00:00
__v: 0
```

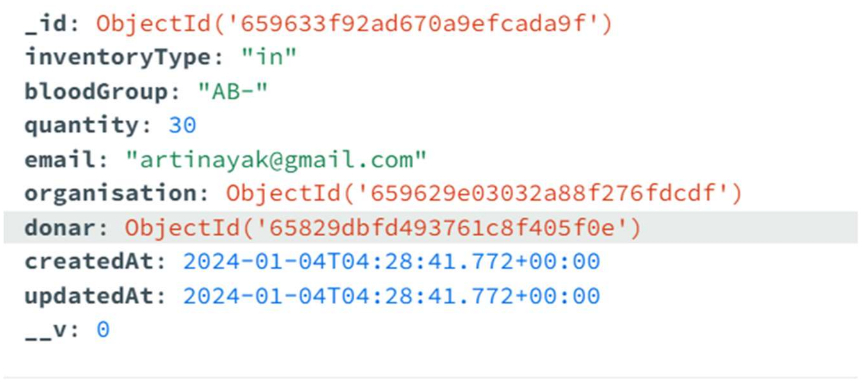
INVENTORY DATABASE



```
_id: ObjectId('65962e403032a88f276fdd1e')
inventoryType: "in"
bloodGroup: "B+"
quantity: 44
email: "saishreepriyadarshini@gmail.com"
organisation: ObjectId('659629e03032a88f276fdcdf')
donar: ObjectId('65962dc43032a88f276fdd14')
createdAt: 2024-01-04T04:04:16.736+00:00
updatedAt: 2024-01-04T04:04:16.736+00:00
__v: 0
```



```
_id: ObjectId('659632407168047712O4e8dc')
inventoryType: "out"
bloodGroup: "B+"
quantity: 30
email: "aiims@gmail.com"
organisation: ObjectId('659629e03032a88f276fdcdf')
hospital: ObjectId('658ae94f6457436c87cf2286')
createdAt: 2024-01-04T04:21:20.972+00:00
updatedAt: 2024-01-04T04:21:20.972+00:00
__v: 0
```



```
_id: ObjectId('659633f92ad670a9efcada9f')
inventoryType: "in"
bloodGroup: "AB-"
quantity: 30
email: "artinayak@gmail.com"
organisation: ObjectId('659629e03032a88f276fdcdf')
donar: ObjectId('65829dbfd493761c8f405f0e')
createdAt: 2024-01-04T04:28:41.772+00:00
updatedAt: 2024-01-04T04:28:41.772+00:00
__v: 0
```

CONCLUSION

In conclusion, Blood Bank Management is a vital and multifaceted discipline that plays a pivotal role in ensuring the availability, safety, and efficient utilization of life-saving blood products. From donor recruitment and engagement to meticulous inventory management, the scope of blood bank management encompasses the entire lifecycle of blood—from donation to transfusion. The implementation of advanced technologies, stringent regulatory compliance, and community-driven initiatives contribute to the successful orchestration of these life-critical processes.

The significance of a well-organized blood bank cannot be overstated, as it directly influences patient care, emergency response capabilities, and public health. The continuous evolution of Blood Bank Management reflects the commitment to innovation, data-driven decision-making, and the pursuit of excellence in healthcare.

As blood banks embrace technological integration, streamlined workflows, and community outreach, they not only meet the immediate demands for blood products but also contribute to long-term public health objectives. Ultimately, effective Blood Bank Management serves as a cornerstone in the healthcare ecosystem, exemplifying the synergy between medical advancements, compassionate donor engagement, and the relentless pursuit of saving lives through safe and accessible blood supplies.

FUTURE ENHANCEMENT

The future enhancements of Blood Bank Management are likely to revolve around leveraging emerging technologies, improving efficiency, and addressing evolving healthcare needs. Some potential future enhancements include:

Blockchain Technology:Implementation of blockchain for enhanced security and transparency in tracking the entire blood supply chain, from donation to transfusion, ensuring data integrity and reducing the risk of errors.

Artificial Intelligence (AI) and Predictive Analytics:Integration of AI and predictive analytics to forecast blood demand, optimize inventory levels, and streamline resource allocation, reducing shortages and minimizing wastage.

IoT and RFID Technology:Increased use of Internet of Things (IoT) and Radio-Frequency Identification (RFID) technology for real-time monitoring of blood products, improving traceability, and minimizing the chances of errors in inventory management.

Mobile Apps and Telehealth Integration:Development of mobile applications to engage donors, schedule appointments, and provide real-time updates on donation campaigns. Integration with telehealth platforms to facilitate remote consultations for donor eligibility assessments.

Automated Testing and Robotics:Automation of laboratory testing processes using robotics to enhance the speed and accuracy of blood testing, reducing manual errors and expediting the screening of donated blood.

Biometric Authentication:Implementation of biometric authentication for secure donor identification, ensuring the confidentiality and integrity of donor information.

Cloud-Based Solutions:Migration to cloud-based solutions for improved accessibility, scalability, and data management, enabling seamless collaboration between different healthcare entities and enhancing disaster recovery capabilities.

Data Interoperability:Focus on data interoperability standards to facilitate seamless integration with electronic health records (EHRs), hospital information systems, and other healthcare platforms, ensuring comprehensive patient care.

Bibliography

the following reference has been used to develop the project “Online Shopping Management Sysetm”: -

Books: -

React Quickly by Azat Marden

React Cookbook

Node.JS Web Development by David Herron

Programming JavaScript Applications Robust Web Architecture with Node, HTML5, and Modern JS Libraries by Eric Elliot

MongoDB: The Definitive Guide 1st Edition by Kristina Chodorow and MichaelDirolf.

Web Source: -

www.w3schools.com

www.wikipedia.org

www.javatpoint.com

www.stackoverflow.com

https://www.mongodb.com/

www.tutorialspoint.com

www.jsp.net

www.javatpoint.com

[1] 1 https://www.w3schools.com

[2] https://www.electionsonline.com/online-voting-system/

[3] https://en.wikipedia.org/wiki/Electronic_voting