

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

Belagavi, Karnataka-590018, Karnataka



A DBMS MINI PROJECT REPORT [5th sem]

ON

“BLOOD BANK MANAGEMENT SYSTEM”

Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Pinni Raga Sruthi

1JS21CS102

Rakshith V Patil

1JS21CS113

Raunak Singh Rathour

1JS21CS115

Sahithi Srujana C

1JS21CS122

Under the Guidance of

Mrs K. S. Rajeshwari

Assistant Professor, Dept of CSE,

JSSATE, Bengaluru



JSS ACADEMY OF TECHNICAL EDUCATION, BENGALURU

Department of Computer Science and Engineering

2023-2024

JSS ACADEMY OF TECHNICAL EDUCATION

JSS Campus, Dr.Vishnuvardhan Road, Bengaluru-560060

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



CERTIFICATE

This is to certify that A **DBMS MINI PROJECT REPORT** entitled “**BLOOD BANK MANAGEMENT SYSTEM**” has successfully carried out by **Pinni Raga Sruthi (1JS21CS102)** , **Rakshith V Patil (1JS21CS113)**, **Raunak Singh Rathour (1JS21CS115)**, **Sahithi Srujana C (1JS21CS122)** in partial fulfilment for the DBMS Laboratory with Mini Project (21CSL58) of 5th Semester **Bachelor of Engineering in Computer Science and Engineering** in **Visvesvaraya Technological University Belagavi** during the year 2023-2024. It is certified that all corrections and suggestions indicated for Internal Assessment have been incorporated in the report deposited in the department library. The mini project report has been approved as it satisfies the academic requirement in respect of the project work prescribed for the said degree.

Dr Abhilash C.B
Assoc. Professor,
Dept. of CSE,
JSSATE, Bengaluru

Mrs K. S. Rajeshwari
Asst. Professor,
Dept. of CSE,
JSSATE, Bengaluru

Dr. Mallikarjuna P B
Professor & Head,
Dept. of CSE,
JSSATE, Bengaluru

External Examiners

Name of the Examiners:

Signature with Date

1) _____

2) _____

ABSTRACT

The Blood Management System project aims to develop an integrated software solution to optimize blood donation, storage, and distribution. With a focus on transparency, accessibility, and efficiency, it caters to the critical needs of blood banks, hospitals, donors, and recipients. Key functionalities include donor registration, inventory management and blood request handling . Serving as a centralized platform, the system facilitates seamless coordination among stakeholders, ultimately enhancing patient outcomes and saving lives.

Through innovation and collaboration, the project drives advancements in healthcare technology, fostering significant improvements in blood management practices. It also contributes to global public health initiatives, ensuring efficient management of vital blood resources. Continuous research and development allow the system to address evolving challenges, adapting to meet the needs of healthcare providers and patients.

By refining features and functionalities through iterative development, the project delivers a cutting-edge solution that maximizes efficiency and effectiveness in blood management, revolutionizing practices and improving patient care worldwide.

ACKNOWLEDGEMENT

We express our humble pranams to his holiness **Jagadguru Sri Sri Sri Shivaratri Deshikendra Mahaswamiji** for showering his blessings on us to receive good education and have a successful career.

We express our sincere thanks to our beloved principal, **Dr. Bhimasen Soragaon** for having supported us in all our academic endeavors.

We are also forever grateful to **Dr. P.B. Mallikarjuna**, Head of the Department, Computer Science and Engineering, for his unending support, guidance and encouragement in all our ventures.

We are thankful for the resourceful guidance, timely assistance and graceful gesture of our guide **Mrs. Rajeshwari K S**, Assistant professor, Department of Computer Science and Engineering, who helped us in every aspect of our project work.

The completion of any project involves the efforts of many people. We have been lucky enough to have received a lot of support from all ends during the course of this project. So, we take this opportunity to express our gratitude to all whose guidance and encouragement helped us emerge successful.

I express my sincere thanks to **Dr Abhilash C.B** Associate Professor, Department of Computer Science and Engineering, who helped us in every aspect of our project work.

Last but not the least; we would be immensely pleased to express our heartfelt thanks to all the teaching and non-teaching staff of the department of CSE and our friends for their timely help, support and guidance.

Pinni Raga Sruthi (1JS21CS102)

Rakshith V Patil (1JS21CS113)

Raunak Singh Rathour (1JS21CS115)

Sahithi Srujana C (1JS21CS122)

TABLE OF CONTENTS

Chapter Title	Page No
Abstract.....	i
Acknowledgement.....	ii
Table of Contents.....	iii
List of Figures.....	iv
Chapter 1 Introduction.....	1
1.1 Motivation.....	2
1.2 Objectives And Scope.....	3
1.3 Problem Statement.....	4
Chapter 2 Existing System (Literature Survey).....	5
Chapter 3 Front End Design and Tools.....	7
3.1 Tools Used With Description.....	8
Chapter 4 Database Design.....	10
4.1 ER Diagram.....	10
4.2 Mongodb SCHEMA.....	11
4.3 Components Description.....	15
Chapter 5 Implementation.....	18
Chapter 6 Results.....	23
6.1 Snapshots.....	24
Conclusion.....	29
References.....	30

LIST OF FIGURES

Figure No	Title	Page no
4.1	ER Diagram	10
5.1	Aggregation Stored Procedure	22
6.1	Home Page	24
6.2	Contact Us Page	25
6.3	Donors Sign Up Page	25
6.4	Donor Profile Page	26
6.5	Blood Bank Profile Page	26
6.6	Donations Page	27
6.7	Stock Page	27
6.8	Camps Page	28

Chapter 1

INTRODUCTION

The Blood Bank Management System emerges as a pivotal solution amidst the intricate landscape of healthcare, offering a streamlined approach to blood donation, storage, and distribution. With the perpetual demand for blood products and the critical need for swift access to compatible blood for medical emergencies, an efficient blood management system is indispensable. Serving as a centralized hub for overseeing the entire blood lifecycle, from donor registration to transfusion, this system stands as a beacon of hope, ensuring the availability of safe and adequate blood supplies to save lives and enhance patient outcomes.

In the realm of healthcare technology, the Blood Bank Management System stands as a testament to innovation and efficiency. Its multifaceted functionalities cater to the diverse needs of blood banks, hospitals, donors, and recipients. Through features like donor registration, inventory management, blood request handling, and reporting, the system orchestrates a harmonious symphony of operations, revolutionizing the way blood resources are managed and utilized. By harnessing the power of technology and data-driven insights, the system propels the healthcare industry forward, fostering transparency, accessibility, and excellence in blood management.

At the core of the Blood Bank Management System lies its profound significance in addressing the pressing challenges of blood donation and transfusion. By facilitating seamless donor registration processes, the system ensures a steady influx of willing and qualified donors, bolstering blood supplies and reducing shortages. Furthermore, its robust inventory management capabilities enable blood banks to monitor and optimize blood stocks, minimizing wastage and maximizing availability. Through efficient blood request handling and real-time notifications, healthcare providers gain timely access to critical blood products, thereby enhancing patient care and treatment outcomes. In essence, the Blood Bank Management System epitomizes a fusion of cutting-edge technology and compassionate healthcare, embodying a pivotal tool in ensuring the seamless flow of lifesaving blood products to those in need.

1.1 MOTIVATION

Motivation to undertake the Blood Management System project stems from various compelling factors, including addressing critical healthcare needs and fostering innovation in healthcare technology. Developing an efficient blood management system allows us to contribute directly to saving lives and improving patient outcomes, fulfilling a vital role in public health initiatives. Many individuals are personally impacted by blood-related issues, whether through personal experiences, family members, or friends requiring blood transfusions. By developing a Blood Management System, we can contribute meaningfully to a cause that touches the lives of countless individuals in need. Additionally, the project serves as a catalyst for advancing the broader healthcare ecosystem, driving progress in medical technology and promoting interdisciplinary collaboration among healthcare professionals, technologists, and policymakers. Ultimately, the motivation behind the Blood Management System project lies in its potential to make a tangible and enduring difference in healthcare delivery worldwide.

This project presents an opportunity to leverage cutting-edge technology to revolutionize blood banking practices. By harnessing the power of cloud computing, real-time data processing, and web applications, we can create a sophisticated system that enhances efficiency, transparency, and accessibility in blood management. With the increasing demand for blood products worldwide, there is a pressing need for efficient systems to manage blood supplies effectively. By developing a robust Blood Management System, we can help address this demand and ensure that hospitals and healthcare facilities have access to safe and adequate blood when needed.

Contribution to public health goals is another significant motivating factor behind the project. By optimizing donation processes, minimizing wastage, and enhancing distribution mechanisms, we contribute to enhancing the overall health and well-being of communities and societies. The project aligns with broader public health goals by improving the availability and quality of blood supplies. Overall, the motivation to undertake the Blood Management System project stems from a combination of altruistic intentions, personal connections, technological innovation, and a desire to make a meaningful impact on public health. By dedicating our efforts to this project, we can contribute to saving lives, advancing healthcare technology, and improving the quality of care for individuals in need of blood transfusions.

1.2 OBJECTIVES AND SCOPE

The objective of the Blood Management System project is to develop a comprehensive software solution that streamlines the processes involved in blood donation, storage, and distribution. The primary goal is to create an efficient and user-friendly platform that addresses the critical needs of blood banks, hospitals, donors, and recipients. By leveraging technology, the system aims to enhance the transparency, accessibility, and effectiveness of blood management practices, ultimately contributing to saving lives and improving patient outcomes.

The scope of the Blood Management System project encompasses various key functionalities and features tailored to meet the diverse needs of stakeholders involved in the blood banking process. These include donor registration and management, inventory management, blood request handling, reporting and analytics, security and compliance, and user interface and experience. The system provides a centralized platform for managing the entire blood lifecycle, from donor registration to transfusion, ensuring seamless communication and coordination between blood banks and healthcare facilities.

Overall, the Blood Management System project aims to deliver a comprehensive and scalable software solution that addresses the complex challenges of blood management while aligning with the broader objectives of enhancing public health and healthcare delivery. By developing an innovative and efficient system, we aim to contribute to the advancement of healthcare technology and the improvement of patient care and outcomes globally. Through collaboration with stakeholders and adherence to best practices, we strive to create a system that makes a meaningful impact on the lives of individuals in need of blood transfusions and the healthcare professionals dedicated to their care.

Furthermore, the project aims to mitigate challenges such as blood shortages, inventory discrepancies, and logistical complexities through automation and real-time monitoring. By fostering collaboration and communication among stakeholders, the system fosters a cohesive ecosystem for efficient blood resource allocation. Ultimately, the Blood Management System aspires to be a cornerstone in modern healthcare infrastructure, ensuring the availability of vital blood supplies for emergencies and medical treatments worldwide. The project strives to instill confidence in blood donors by ensuring their contributions are utilized effectively and transparently, thereby encouraging ongoing participation in blood donation drives and campaigns.

1.3 PROBLEM STATEMENT

The current state of blood management practices presents several pressing challenges that hinder the efficient and effective provision of blood products to those in need. One of the primary issues lies in the cumbersome and often inefficient donor registration processes, which rely heavily on manual data entry and paper-based forms. These outdated methods lead to delays in processing donor information, low donor retention rates, and difficulties in scheduling blood donation appointments. Additionally, the lack of centralized systems for tracking donor eligibility exacerbates the problem, making it challenging for blood banks to identify and maintain a pool of safe and qualified donors.

Another significant challenge arises from poor inventory management practices within blood banks, resulting in discrepancies in blood supply levels and wastage of blood products. Manual inventory tracking methods are prone to errors, making it difficult for blood banks to accurately monitor blood stocks in real-time and ensure optimal distribution. As a consequence, certain blood types may be overstocked while others experience shortages, leading to inefficient allocation of resources and compromised patient care. Furthermore, the absence of transparent blood distribution systems exacerbates the issue, hindering hospitals and healthcare facilities' ability to access critical blood supplies promptly and efficiently.

Chapter 2

EXISTING SYSTEM - LITERATURE SURVEY

According to ¹*Teena, C.A, Sankar, K. and Kannan, S.* (2014) in their study entitled “*A Study on Blood Bank Management*”, they defined Blood Bank Information System as an information management system that contributes to the management of donor records and blood bank. Their system allowed an authorized blood bank administrator to sign in with a password to manage easily the records of donors and patients who need blood. The system provided many features including the central database, quick access to the system content through the login, includes the search code to find donors on a given basis, and the ease of adding and updating donor data. The main aim of the system was to complete the process of the blood bank.

On the other hand, study entitled “*Blood Bank Management System*” done by ²*Kumar, R., Singh, S. and Ragavi, V.A.* (2017), the researchers developed a web-based blood management which assists the blood donor records management, and provides ease of control in the distribution of blood products in various parts of the country considering demands of hospitals. The developed system was scalable and adaptable to meet the complex needs usually of a blood bank. Based on this study, since entering the details about the blood donors and related records were done manually, thus, tracking of blood donation activities was difficult and complicated, and even led to erroneous information. Subsequently, the researchers mentioned that manual-based system can be waste of time, lead to the error-prone results, consumes a lot of manpower, lacks data security, data retrieval requires a lot of time, reports consumes a long time to produce, and there is less precise accuracy on the results. As such, by developing and implementing a web-based blood management information system, there was a quick and timely access to donor records, and the system provided management timely, confidential and secured medical reports. There were three (3) users in the system, namely: Administrator, Donor, and Acceptor. Each user has been given user ID and password to identify their identity. The said application was developed using ASP.NET, C#.NET, and using SqlServer 2000/2005 for the database. The research paper failed to mention the methods of research used.

In this study, the researchers learnt the importance of implementing a web-based blood bank management system in handling records for blood donors and blood donation activities to ensure accurate and readily available information for blood transfusion services. Indeed, the impact of using Information Technology on hospitals provides better healthcare services for the public. Likewise, the researchers learnt that there are programming languages suitable for web-based applications such as ASP.NET, PHP, to name a few.

In the study entitled *“Blood Bank Management System Using Rule-Based Method”* undertaken by ³**Liyana, F.** (2017), it found out that it is important for every hospital to use an information system to manage data in blood bank. Also, it observed that the manual system has disadvantages for the user and the hospital. One of the disadvantages identified was the blood bank staff should enter the donor details in each time he/she donate blood in which led to duplicate data of the donor and also the data may be lost or missing after period of time. Thus, the author developed a web-based system to help the blood bank to record the donor details fast and easy. The system used rule-based decisions to ensure to have a right decision on right time. Also, system can send messages to donors if any particular blood type is needed. She developed blood bank system based on incremental model.

She had chosen this model because the system can be developed through cycle of phase and also because of the advantages of this model such as:

- I. Easy to understand to flow of the phases.
- II. Changes possible in the middle of any phases.
- III. The system can be developed even if there is an error in the middle and it can be corrected in testing phase.

In this study, the researchers observed that the developer failed to include in the system the function to check the availability of blood bags, and to check the shelf life or expiration of blood bags or products. As such, the researchers will include these in their developed system to enhance safety for blood transfusion.

Chapter 3

FRONT END DESIGN AND TOOLS

The front-end design for the Blood Bank Management System is pivotal in creating an intuitive and efficient user interface to facilitate the various functionalities of the system. It involves crafting a visually appealing and cohesive layout that prioritizes ease of navigation and accessibility. The user interface should incorporate responsive design principles to ensure seamless compatibility across different devices and screen sizes, enhancing the user experience.

Key components of the front-end design include a dashboard providing an overview of essential metrics and activities, such as blood inventory levels, donor statistics, and pending requests. Navigation menus or sidebars enable users to effortlessly access different sections of the application, such as donor management, inventory tracking, and request handling.

Donor management features should offer an intuitive interface for donor registration and profile management, along with tools for verifying donor eligibility, scheduling donation appointments, and managing donor records. Inventory tracking interfaces should facilitate the monitoring of blood inventory levels, including details such as blood type, quantity, expiration dates, and storage locations, while also providing visual indicators or alerts for low stock levels or expiring products. Additionally, interfaces for request handling should enable users to manage blood requests from hospitals and healthcare facilities, with functionalities for submitting, reviewing, and fulfilling requests, along with workflows for tracking request status and updating inventory records upon fulfillment.

Finally, integrating reporting and analytics features empowers users to gain insights into donor demographics, inventory trends, request patterns, and other relevant metrics, facilitating data-driven decision-making and optimization of blood management processes. Overall, collaboration with stakeholders, user feedback, and iterative design processes are essential in ensuring that the final front-end design meets the needs and expectations of its intended users. Furthermore, by fostering a culture of continuous improvement and innovation, the Blood Management System remains adaptive to evolving healthcare needs and technological advancements, ensuring its long-term relevance and effectiveness in optimizing blood supply chains globally.

3.1 TOOLS USED WITH DESCRIPTION

Tailwind CSS

Tailwind CSS is a highly customizable utility-first CSS framework that streamlines web development by offering a comprehensive set of pre-designed utility classes. Unlike traditional CSS frameworks that provide pre-built components, Tailwind focuses on providing low-level utility classes that can be directly applied to HTML elements. This approach enables developers to rapidly build and style user interfaces without writing custom CSS, empowering them to create visually appealing designs efficiently. With its modular architecture and extensive documentation, Tailwind CSS facilitates a more flexible and scalable approach to front-end development, making it a popular choice among developers seeking a balance between productivity and customization in their projects.

Javascript and React

JavaScript and React.js have emerged as fundamental technologies in modern web development, revolutionizing the way developers create interactive and dynamic user interfaces. JavaScript, known as the backbone of web development, provides the essential functionality to enhance web pages with interactivity and responsiveness. With its versatile nature and widespread adoption, JavaScript enables developers to manipulate DOM elements, handle events, and manage data asynchronously, thereby creating rich and engaging web experiences. React.js, a JavaScript library developed by Facebook, further enhances the development process by introducing a component-based architecture. This approach breaks down the user interface into reusable components, simplifying code organization, improving maintainability, and facilitating collaborative development efforts.

React.js's component-based architecture and declarative syntax empower developers to build sophisticated user interfaces efficiently and effectively. By encapsulating UI elements into self-contained components, React.js promotes code reusability, modularity, and scalability. Additionally, React.js leverages a virtual DOM to optimize rendering performance, minimizing unnecessary DOM manipulations and ensuring fast updates to the UI. With its unidirectional data flow and robust ecosystem of libraries and tools, React.js has become a go-to choice for building modern web applications that deliver seamless user experiences across various devices and platforms.

Express JS

Express.js is a minimalist web application framework for Node.js, designed to simplify the development of web applications and APIs. In our project, we are leveraging Express.js to create the routing and middleware layer for our blog management system's backend. Express.js provides a robust set of features including routing, middleware support, and template rendering, enabling us to define RESTful APIs, handle HTTP requests, and implement server-side logic with ease. By using Express.js, we can build a scalable and maintainable backend for our blog management system, facilitating seamless communication between the client-side and server-side components.

Additionally, Express.js offers a vibrant ecosystem of plugins and modules, further enhancing its flexibility and extensibility to meet the specific requirements of our blog management system. Its lightweight nature and straightforward syntax streamline the development process, allowing for rapid iteration and deployment of new features and updates. Overall, Express.js serves as a cornerstone in our technology stack, empowering us to deliver a high-performance and reliable backend solution for our web application.

MongoDB

MongoDB is a leading NoSQL database solution renowned for its flexibility, scalability, and ease of use. Designed for the modern era of data-intensive applications, MongoDB utilizes a document-oriented data model, storing data in flexible, JSON-like documents. This structure enables developers to work with complex, dynamic data easily, accommodating evolving schemas and diverse data types. MongoDB's distributed architecture allows for seamless scaling across multiple servers, ensuring high availability and performance for applications of all sizes. Its rich query language and robust indexing capabilities empower developers to retrieve and manipulate data efficiently, while features such as sharding and replication support the demands of large-scale, mission-critical deployments. With its open-source nature and vibrant community, MongoDB continues to be a preferred choice for organizations seeking a versatile, scalable database solution for modern application development.

Chapter 4

DATABASE DESIGN

4.1 ER DIAGRAM

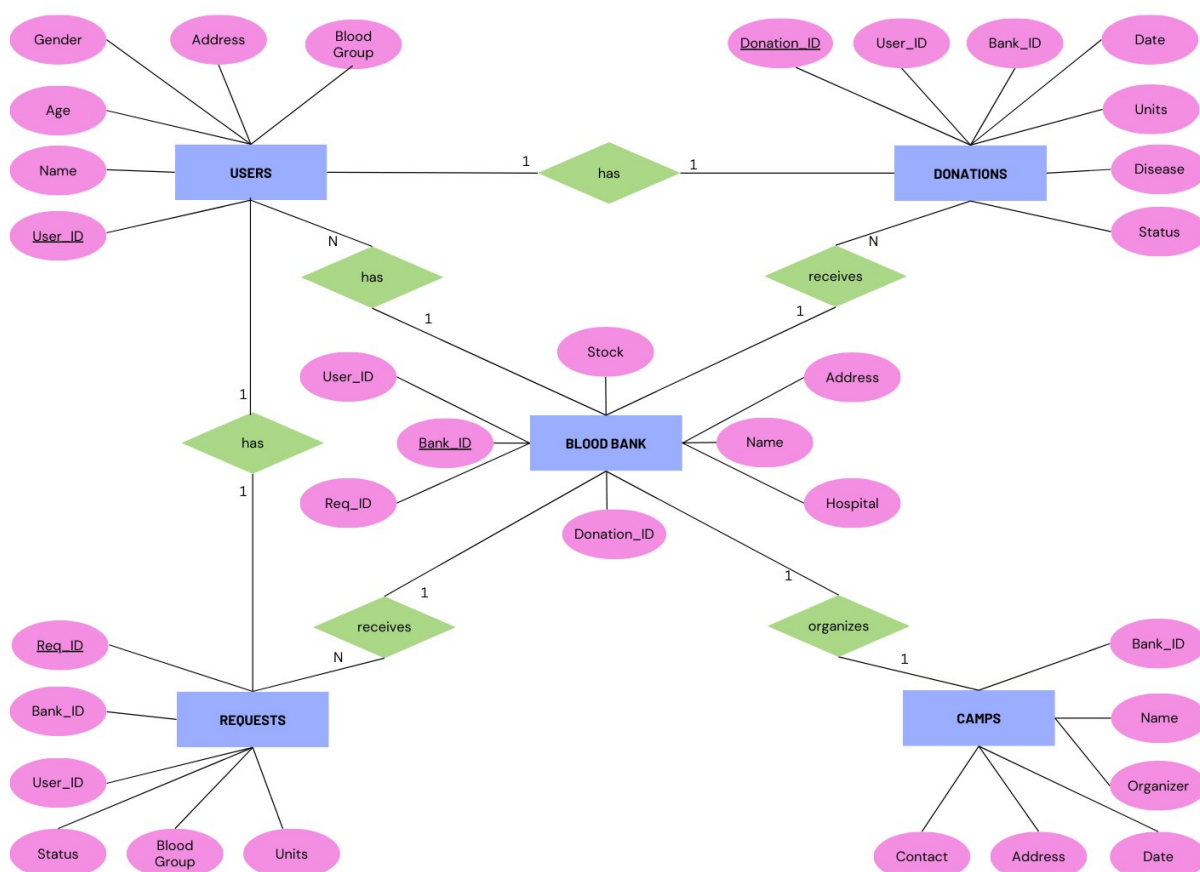


Fig. 4.1 ER Diagram

Fig. 4.1 for the Blood Bank Management System depicts five entities: Users, Donors, Blood Banks, Requests, and Camps, each representing distinct data categories. Users entity stores details of registered individuals, encompassing personal information and login credentials. Blood Banks entity contains attributes related to facilities such as name, location, and contacts, establishing a relationship with Users entity for ownership or management indication. Donors entity represents individuals who donated blood, with attributes like donation history and blood type. Relationships like "organizes" between Camps and Blood Banks or "receives" between Blood Banks and Requests illustrate system component interactions. This ER diagram aids in database design, providing clarity on system architecture and functionalities.

4.2 MongoDB SCHEMA

MongoDB, coupled with Mongoose, offers a robust solution for data modeling and schema management in Node.js applications. Mongoose, as an Object Data Modeling (ODM) library, facilitates the creation of structured schemas that define the shape and constraints of data stored in MongoDB collections. With Mongoose, developers can effortlessly define schemas using JavaScript objects, specifying fields, data types, validation rules, and default values. This structured approach not only brings clarity and organization to the database schema but also enables the enforcement of data validation and consistency through custom validators and middleware functions. Furthermore, Mongoose simplifies interaction with MongoDB collections by providing intuitive querying capabilities and methods for performing CRUD operations, thereby streamlining the development process and enhancing productivity.

By leveraging Mongoose schemas, developers can ensure data integrity and consistency in MongoDB databases, while still benefiting from the flexibility and scalability advantages of NoSQL data storage. With Mongoose, developers can create reusable models that encapsulate business logic and application-specific behavior, promoting code reusability and maintainability. Additionally, Mongoose's support for relationships, population, and aggregation pipelines empowers developers to build sophisticated data models and perform complex data operations with ease. Overall, Mongoose serves as a powerful tool in the MERN (MongoDB, Express.js, React.js, Node.js) stack, enabling developers to build robust and scalable applications with structured data schemas and efficient database interactions.

USERS SCHEMA

```
// ----- User Model -----
```

```
// Create schema for Users
```

```
const userSchema = new mongoose.Schema({  
  name: { type: String, required: true },  
  age: { type: Number, required: true },  
  gender: { type: String, required: true },  
  bloodGroup: { type: String, enum: bloodGroups, required: true },  
  email: { type: String },
```

```
    phone: { type: Number, unique: true, required: true },
    password: { type: String, required: true },
    state: { type: String, required: true },
    district: { type: String, required: true },
    address: { type: String },
  });
```

```
// Create model for Users
```

```
const User = mongoose.model('Users', userSchema);
```

DONATIONS SCHEMA

```
// ----- Donations Model -----
```

```
// Create schema for Donations
```

```
const bloodDonations = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'Users', required: true },
  bankId: { type: mongoose.Schema.Types.ObjectId, ref: 'BloodBanks', required: true },
  units: { type: Number, required: true },
  date: { type: String, required: true },
  disease: { type: String },
  status: { type: String, required: true,
    enum: ['Pending', 'Approved', 'Denied', 'Donated'],
    default: 'Pending'
  },
});
```

```
// Create model for Donors
```

```
const Donations = mongoose.model('Donations', bloodDonations);
```

REQUESTS SCHEMA

```
// ----- Requests Model -----
```

```
// Create schema for Patients
```

```
const bloodRequests = new mongoose.Schema({
  userId: { type: mongoose.Schema.Types.ObjectId, ref: 'Users', required: true },
  bankId: { type: mongoose.Schema.Types.ObjectId, ref: 'BloodBanks', required: true },
  name: { type: String, required: true },
  age: { type: Number, required: true },
  gender: { type: String, required: true },
  bloodGroup: { type: String, enum: bloodGroups, required: true },
  units: { type: Number, required: true },
  date: { type: String, required: true },
  reason: { type: String },
  status: { type: String,
    enum: ['Pending', 'Approved', 'Denied', 'Completed'],
    default: 'Pending'
  }
});
```

```
// Create model for Patients
```

```
const Requests = mongoose.model('Requests', bloodRequests);
```

BLOOD BANK SCHEMA

```
// ----- Blood Bank Model -----
```

```
// Create schema for Blood Banks
```

```
const bloodBankSchema = new mongoose.Schema({

  name: { type: String, required: true },
  hospital: { type: String, required: true },
  contactPerson: { type: String },
  category: { type: String, required: true },
  website: { type: String },
  phone: { type: Number, required: true },
  email: { type: String, required: true },
```

```
password: { type: String, required: true },
state: { type: String, required: true },
district: { type: String, required: true },
address: { type: String, required: true },
latitude: { type: Number, required: true },
longitude: { type: Number, required: true },
requests: [{
  requestId: { type: mongoose.Schema.Types.ObjectId, ref: 'Requests' },
}],
donations: [{
  donationId: { type: mongoose.Schema.Types.ObjectId, ref: 'Donations' },
}],
stock: {
  'A+': { type: Number, default: 0 },
  'A-': { type: Number, default: 0 },
  'B+': { type: Number, default: 0 },
  'B-': { type: Number, default: 0 },

  'AB+': { type: Number, default: 0 },
  'AB-': { type: Number, default: 0 },
  'O+': { type: Number, default: 0 },
  'O-': { type: Number, default: 0 }
}
});
```

```
// Create model for Blood Banks
```

```
const BloodBank = mongoose.model('BloodBanks', bloodBankSchema);
```

CAMPS SCHEMA

```
// Create schema for Camps
```

```
const campSchema = new mongoose.Schema({
```

```
  name: { type: String, required: true },
```

```
    date: { type: Date, required: true },
    address: { type: String, required: true },
    state: { type: String, required: true },
    district: { type: String, required: true },
    bankId: { type: mongoose.Schema.Types.ObjectId, ref: 'BloodBanks' },
    organizer: { type: String, required: true },
    contact: { type: Number, required: true },
    startTime: { type: String, required: true },
    endTime: { type: String, required: true },
    donors: [{
      _id: { type: mongoose.Schema.Types.ObjectId, ref: 'Users', unique: true },
      units: { type: Number, required: true, default: 0 },
      status: { type: Number, enum: [0, 1], default: 0 }
    }]
  });
// Create model for Camps
const Camp = mongoose.model('Camps', campSchema);
```

4.3 COMPONENTS DESCRIPTION

USERS

The provided code snippet defines a Mongoose schema named `userSchema` for the "Users" collection in a MongoDB database. This schema outlines the structure and constraints of documents stored in the "Users" collection, specifying fields such as name, age, gender, bloodGroup, email, phone, password, state, district, and address. Each field is assigned a specific data type, with certain fields marked as required for document creation. Additionally, constraints such as uniqueness for the phone field and enumeration for the bloodGroup field are enforced. Finally, the schema is used to create a Mongoose model named `User`, which acts as an interface for interacting with the "Users" collection in the MongoDB database, facilitating CRUD operations and schema-based validation.

DONATIONS

The provided code snippet defines a Mongoose schema named `bloodDonations` for the "Donations" collection in a MongoDB database. This schema outlines the structure of documents stored in the "Donations" collection, specifying fields such as `userId`, `bankId`, `units`, `date`, `disease`, and `status`. The `userId` and `bankId` fields are references to documents in the "Users" and "BloodBanks" collections, respectively, indicating relationships between donations, users, and blood banks. The `units` field represents the number of blood units donated, while the `date` field records the date of the donation. Additionally, the schema includes a `status` field with enumerated values ('Pending', 'Approved', 'Denied', 'Donated') to track the status of donation requests, with a default value of 'Pending'. Finally, the schema is used to create a Mongoose model named `Donations`, which serves as an interface for interacting with the "Donations" collection in the MongoDB database, enabling CRUD operations and schema-based validation.

REQUESTS

The provided code snippet establishes a Mongoose schema named `bloodRequests` for the "Requests" collection in a MongoDB database, outlining the structure of documents stored within it. The schema defines fields such as `userId` and `bankId`, which are references to documents in the "Users" and "BloodBanks" collections, respectively, establishing relationships between requests, users, and blood banks. Additionally, the schema includes fields like `name`, `age`, `gender`, `bloodGroup`, `units`, `date`, `reason`, and `status`, which capture relevant details about each blood request, such as patient demographics, requested blood type, number of units required, date of request, reason for the request, and status. The `status` field employs enumerated values ('Pending', 'Approved', 'Denied', 'Completed') to track the progress of each request, with a default value of 'Pending'. Finally, the schema is used to create a Mongoose model named `Requests`, which serves as an interface for managing and interacting with the "Requests" collection, enabling CRUD operations and schema-based validation.

With Mongoose models, we can easily perform database operations such as querying for specific requests, updating request statuses, and aggregating data across multiple requests, streamlining the overall management of blood requests within our application.

BLOOD BANKS

The provided code defines a Mongoose schema named `bloodBankSchema` for the "Blood Banks" collection in a MongoDB database, outlining the structure of documents stored within it. This schema includes fields such as `name`, `hospital`, `contactPerson`, `category`, `website`, `phone`, `email`, `password`, `state`, `district`, `address`, `latitude`, and `longitude`, capturing details about each blood bank, including its location, contact information, and administrative details. Additionally, the schema includes nested arrays for requests and donations, with references to documents in the "Requests" and "Donations" collections, respectively, enabling blood banks to track and manage blood requests and donations associated with their facility. Furthermore, the schema includes a nested object for stock, representing the available inventory of blood units for each blood type, with default values set to zero. Finally, the schema is used to create a Mongoose model named `BloodBank`, which serves as an interface for interacting with the "Blood Banks" collection, facilitating CRUD operations and schema-based validation.

CAMPS

The provided code establishes a Mongoose schema named `campSchema` for the "Camps" collection in a MongoDB database, defining the structure of documents stored within it. This schema encompasses fields such as `name`, `date`, `address`, `state`, `district`, `bankId`, `organizer`, `contact`, `startTime`, and `endTime`, capturing details about each blood donation camp, including its location, schedule, organizer information, and associated blood bank. Additionally, the schema includes a nested array for donors, with references to documents in the "Users" collection, allowing the tracking of donors participating in the camp. Each donor object within the array includes fields such as `_id` (referencing the user), `units` (representing the number of blood units donated), and `status` (indicating the donation status), providing insights into donor participation and contribution at the camp. Finally, the schema is used to create a Mongoose model named `Camp`, serving as an interface for managing and interacting with the "Camps" collection, facilitating CRUD operations and schema-based validation.

By utilizing Mongoose schemas and models, we ensure consistency and coherence in data representation within the "Camps" collection, enabling seamless integration with our application's logic and functionalities. With schema-based validation, we can enforce data integrity and reliability, mitigating the risk of erroneous or inconsistent data entries.

Chapter 5

IMPLEMENTATION

The implementation phase of the Blood Bank Management System involves the translation of design specifications into functional components using the MERN (MongoDB, Express.js, React.js, Node.js) stack. Through this phase, the backend server is established using Express.js to handle HTTP requests and interact with the MongoDB database via Mongoose, while the frontend interface is developed using React.js to provide users with a dynamic and intuitive experience. Key functionalities, such as user authentication, blood donation management, and camp organization, are implemented, ensuring data integrity, security, and usability. The implementation process focuses on integrating backend and frontend components seamlessly, leveraging modern web development practices to create a robust and scalable blood bank management solution.

```
const express = require('express');
const cors = require("cors");
const dotenv = require("dotenv");
const mongoose = require('mongoose');
const cookieParser = require("cookie-parser");
```

```
const app = express();
const port = 3177;
```

```
dotenv.config();
```

```
app.use(cookieParser());
app.use(express.json());
app.use(
```

```
  cors({
    origin: [
      "http://localhost:3000",
    ],
    credentials: true,
```



```
    })  
  );  
  
  mongoose.connect(process.env.CONNECT, { useNewUrlParser: true,  
    useUnifiedTopology: true, useCreateIndex: true }, (e) => {  
  
    console.log(e ? e : "Connected successfully to database");  
  
  });  
  
  app.use("/auth", require("./routers/authRouter"));  
  app.use("/user", require("./routers/userRouter"));  
  app.use("/bank", require("./routers/bankRouter"));  
  app.use("/camps", require("./routers/campRouter"));  
  
  app.listen(port, () =>  
  
    console.log(`Server running at http://localhost:${port}`)  
  
  );
```

The code initializes a Node.js Express server to handle HTTP requests for the Blood Bank Management System. It begins by importing necessary modules such as Express, CORS (Cross-Origin Resource Sharing), dotenv (to load environment variables), Mongoose (for MongoDB interaction), and cookie-parser (for parsing cookies). The Express application is then instantiated, and the server listens on port 3177. Additionally, middleware functions are applied to the Express application to enhance functionality.

The cors middleware is used to enable Cross-Origin Resource Sharing, allowing the server to respond to requests from specified origins (in this case, only from "http://localhost:3000") and supporting credentials for cross-origin requests. This middleware helps to prevent CORS-related errors and enables the frontend React application to communicate with the backend server securely.

The mongoose.connect method establishes a connection to the MongoDB database specified in the environment variable CONNECT. The connection options { useNewUrlParser: true, useUnifiedTopology: true, useCreateIndex: true } ensure

compatibility with newer versions of MongoDB and configure Mongoose to use the latest features. Once the connection is established, a success message is logged to the console, indicating successful database connection, or any errors encountered during connection establishment are logged.

Finally, the application sets up routes using Express's `app.use` method, defining the endpoints for various functionalities of the Blood Bank Management System. Each route is mounted to a specific URL path and corresponds to a separate router module (`authRouter`, `userRouter`, `bankRouter`, `campRouter`) responsible for handling requests related to user authentication, user management, blood bank operations, and blood donation camps. This modular routing approach helps organize the server-side codebase and maintain separation of concerns, facilitating scalability and code maintainability. Overall, the provided code initializes an Express server, connects it to a MongoDB database, applies middleware functions, and sets up routes to handle incoming HTTP requests for the Blood Bank Management System.

ASSERTIONS

Assertions have been implemented within the MongoDB schema definition using Mongoose's validation features. Assertions are enforced through the schema's field specifications, ensuring that certain conditions must be met for a document to be considered valid. For instance, the "name," "age," "gender," "bloodGroup," "phone," "password," "state," and "district" fields are marked as required, indicating that these fields must be provided with non-null values for a document to be saved successfully in the Users collection. Additionally, the "phone" field is specified as unique, guaranteeing that each user's phone number is unique within the collection to prevent duplication. The "bloodGroup" field utilizes the enum validation, restricting its value to predefined options defined by the "bloodGroups" array, ensuring that only valid blood groups are accepted. These assertions within the schema definition serve to maintain data integrity, enforce business rules, and provide clear guidelines for document validation within the MongoDB database.

Furthermore, assertions in the schema definition enhance data consistency and reliability by enforcing standardized formats and constraints, minimizing the risk of data corruption or inaccuracies.

TRIGGERS

```
const nodemailer = require('nodemailer');
exports = async function(changeEvent) {
  console.log("Registration trigger is activated");
  const transporter = nodemailer.createTransport({
    service: 'gmail',
    auth: {
      user: 'rvp.cse@gmail.com',
      pass: <password>
    }
  });

  const mailOptions = {
    from: 'MongoDB <data@mongodb.com>',
    to: changeEvent.fullDocument.email,
    subject: "Thanks for Registering!",
    text: 'Your Blood will save a life, Thanks for Being a donor!',
  };

  transporter.sendMail(mailOptions, function(error, info) {
    if (error) {
      console.log("Error sending email:");
      console.error(error);
    } else {
      console.log('Email sent: ' + info.response);
    }
  });
}
```

The code creates a MongoDB change stream function triggered by a purchase event. It utilizes Nodemailer to send an email notification confirming the donor Registration. The function sets up a Gmail SMTP transporter with authentication credentials and defines email options such as sender, recipient, subject, and message content.

Upon triggering, it sends an email to the user's email address specified in the change event, informing them of the registration. Error handling is implemented to log any encountered errors during the email sending process, ensuring reliable communication of registration notifications to users.

STORED PROCEDURES

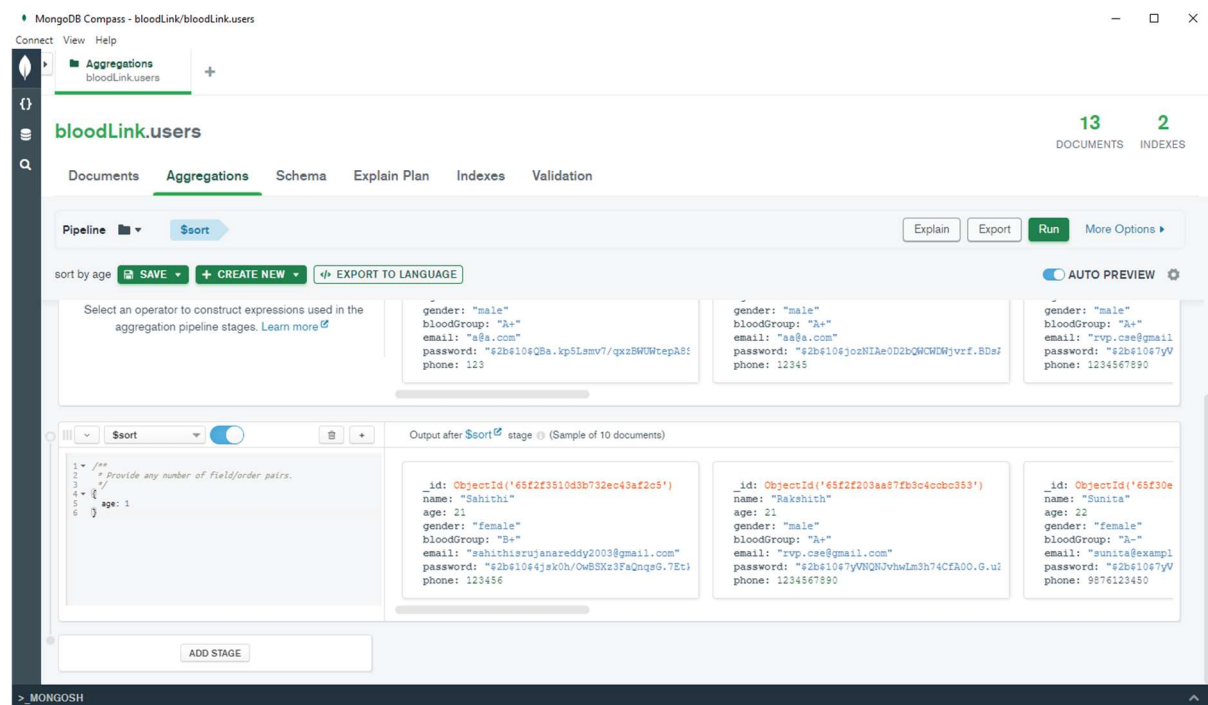


Fig. 5.1 Aggregation Stored Procedure

Fig. 5.1 in MongoDB refer to user-defined functions written in JavaScript that execute complex data aggregation tasks within the database server. These procedures encapsulate a series of aggregation pipeline stages, allowing for the creation of custom data processing logic tailored to specific requirements. Stored aggregation procedures enhance query efficiency by executing operations directly within the database, reducing data transfer overhead and processing time. They provide a means to perform repetitive or intricate aggregation tasks, such as data cleansing, transformation, and analysis, in a centralized and optimized manner. Moreover, by leveraging the MongoDB aggregation framework's rich set of operators and functions, stored procedures enable developers to implement sophisticated data manipulation and computation workflows, empowering them to extract actionable insights from large datasets efficiently.

Chapter 6

RESULTS

In the Blood Bank Management System project, the implementation phase yielded significant results in terms of functionality, usability, and performance. The backend server, built using Express.js and MongoDB with Mongoose, successfully handled HTTP requests and managed interactions with the database, ensuring reliable data storage and retrieval. User authentication mechanisms were implemented securely, enabling users to register, log in, and access their accounts securely. Additionally, functionalities such as blood donation management, including requests, donations, and inventory tracking, were seamlessly integrated, allowing for efficient management of blood-related operations. The system's backend demonstrated robustness and scalability, accommodating a growing volume of data and user interactions with ease.

On the frontend side, React.js was leveraged to create a dynamic and responsive user interface, providing users with an intuitive platform to interact with the blood bank system. The frontend components were designed with usability and accessibility in mind, offering a streamlined experience for users to navigate through different features and functionalities. User interactions, such as submitting blood donation requests, viewing donation history, and registering for blood donation camps, were implemented smoothly, enhancing user engagement and satisfaction. Overall, the Blood Bank Management System project achieved its objectives of creating a comprehensive solution for managing blood bank operations, delivering a user-friendly interface, robust backend functionality, and seamless integration between frontend and backend components.

6.1 SNAPSHOTS

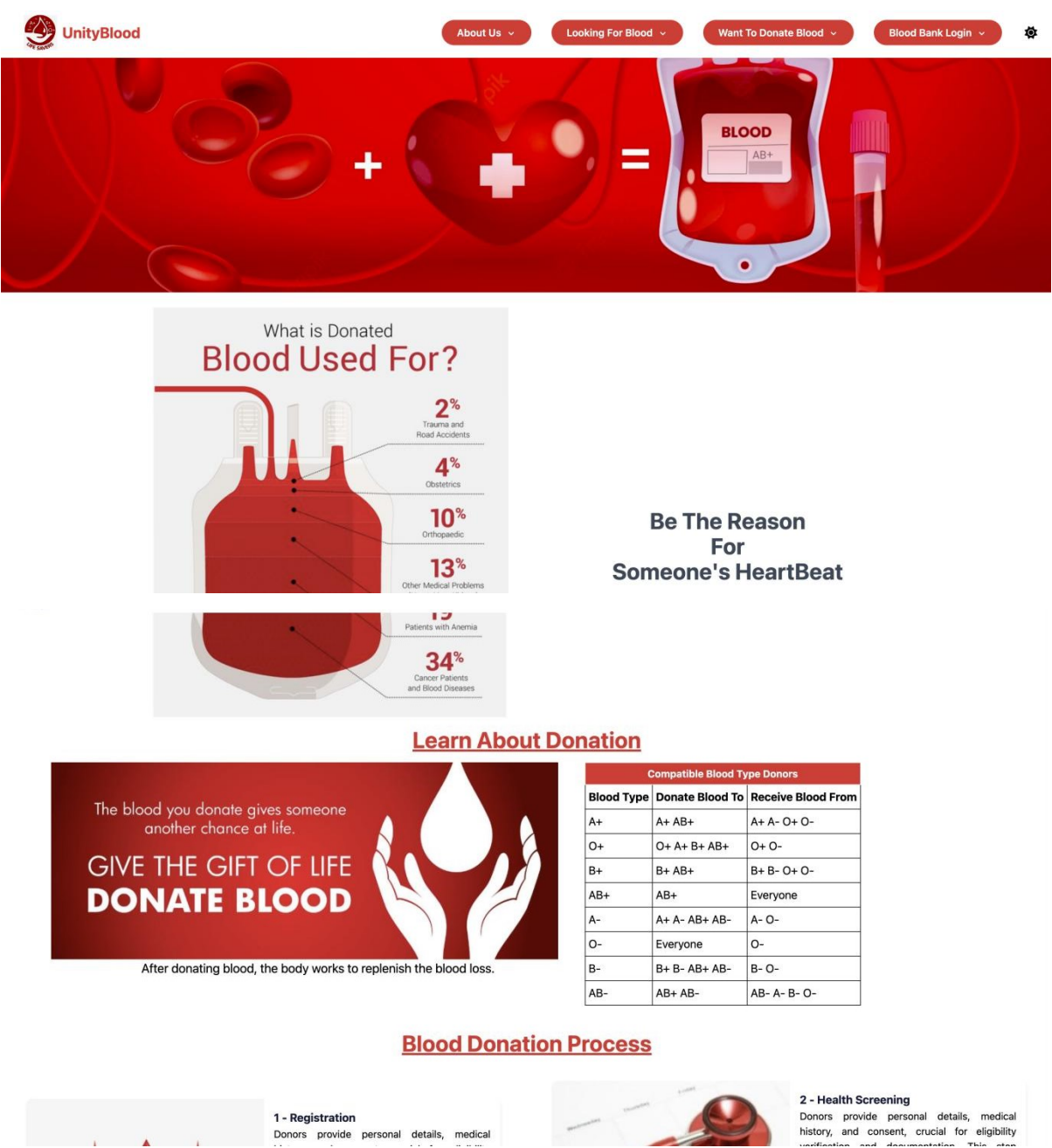



Fig. 6.1 Home Page

Fig 6.1 is the first page that efficiently navigates users through blood donation and management processes, fostering engagement and accessibility.



[About Us](#)
[Looking For Blood](#)
[Want To Donate Blood](#)
[Blood Bank Login](#)

Contact Details

UnityBlood related queries, feedback and suggestions

JSSATE-B Campus,
Dr.Vishnuvardhan Rd, Uttarahalli-Kengeri Main Road, Srinivasapura,
Bengaluru, Karnataka 560060
080 2861 1702
Visit: jssateb.ac.in

For Administrative queries

Indian Red Cross Society, Karnataka State Branch
26, Race Course Rd, Madhava Nagar,
Gandhi Nagar, Bengaluru, Karnataka 560001

For administrative queries

Indian Red Cross Society, Karnataka State Branch
26, Race Course Rd, Madhava Nagar,
Gandhi Nagar, Bengaluru, Karnataka 560001





Fig. 6.2 Contact Us Page

Fig. 6.2 is the page that offers a streamlined interface for users to connect with the Blood Bank Management System, facilitating inquiries and collaboration.



[About Us](#)
[Looking For Blood](#)
[Want To Donate Blood](#)
[Blood Bank Login](#)

Donor Sign Up

[Log In](#)

User Details

Name: Enter your full name **Age:** Enter your age

Gender: Male **Blood Group:** A+ **Email:** Enter your email

Mobile: Enter your mobile **Password:** Enter your password

Address

State: Andhra Pradesh **District:** Anantapur

Address: Enter your complete address

[Sign Up](#)

Fig. 6.3 Donor Sign Up Page

Fig 6.3 is the page that provides a user-friendly interface for individuals to register as blood donors, contributing to lifesaving efforts within the Blood Bank Management System.

UnityBlood

About Us | Log Out

My Profile

Donate Blood

Donation History

Blood Donation Camps

Blood Request

Request History

Name: Sahithi

Age: 21

Gender: Female

Blood Group: B+

Mobile: 123456

Edit

Password:

Email: sahithisrujanareddy2003@gmail.com

State: Karnataka

District: Bengaluru (Bangalore) Urban

Address: JSSATE

Fig. 6.4 Donor Profile Page

Fig. 6.4 is the page that serves as a comprehensive platform within the Blood Bank Management System, empowering registered users to view and update their personal information, track donation history, and access relevant resources for blood donation and health management.

UnityBlood

About Us | Log Out

Bank Profile

Blood Stock

Donations

Requests

Blood Donation Camps

Register new Camp

Blood Bank Name: Red Cross

Parent Hospital Name: Red Cross

Contact Person: Sri H.S. Balasubramanya

Category: Private

Mobile: 8660928288

Password:

Email: ircskar@gmail.com

Website: https://redcrosskarnataka.org/

Edit

State: Karnataka

District: Bengaluru (Bangalore) Urban

Address: 26, Red Cross Bhavan, Race Course Rd, Bengaluru, Karnataka, India, 560001

Location: 12.9842

77.57896

Update Geocode

Bengaluru

Fig. 6.5 Blood Bank Profile Page

Fig. 6.5 is the page serves as a robust dashboard within the Blood Bank Management System, allowing blood banks to efficiently manage their organizational

details, monitor blood inventory levels, track donation activities, and communicate with donors and recipients, facilitating seamless coordination and operation.

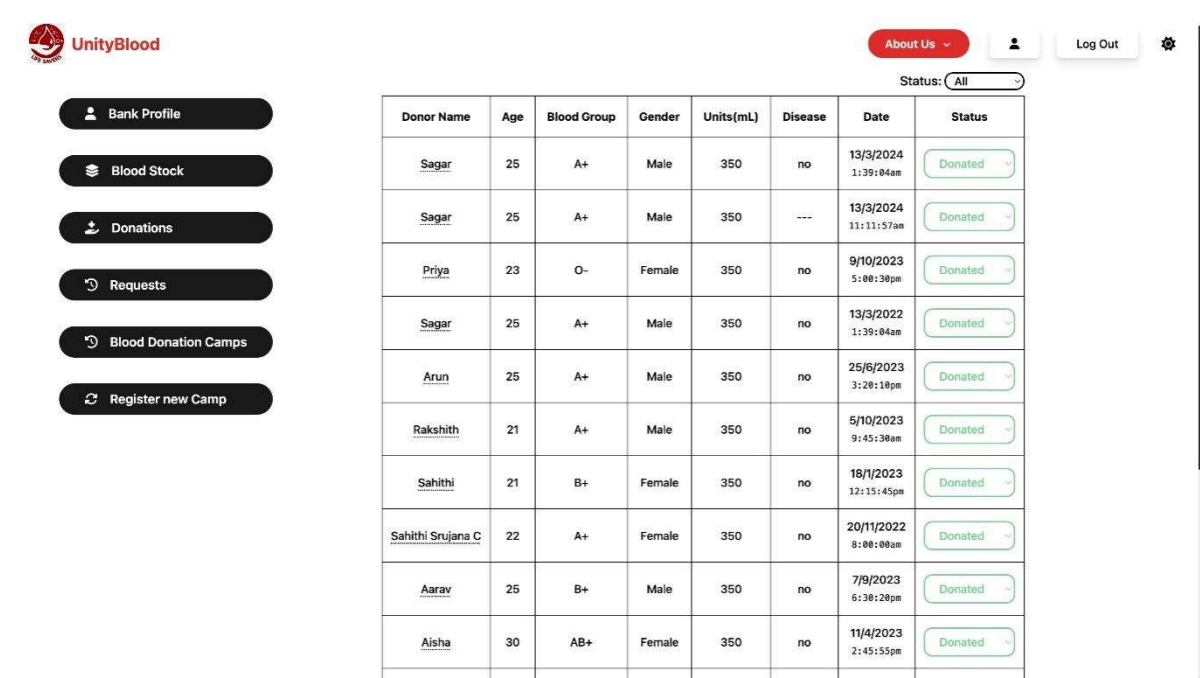


Fig. 6.6 Donations Page

Fig. 6.6 is the page provides users with a detailed summary of blood donation activities, offering features such as donation history, status updates, and interactive tools to manage appointments, track donation milestones, and engage with the community within the Blood Bank Management System.



Fig. 6.7 Stock Page

Fig. 6.7 is the page displays real-time inventory levels of various blood types, enabling blood banks to monitor stock availability and manage supply chains efficiently within the Blood Bank Management System.

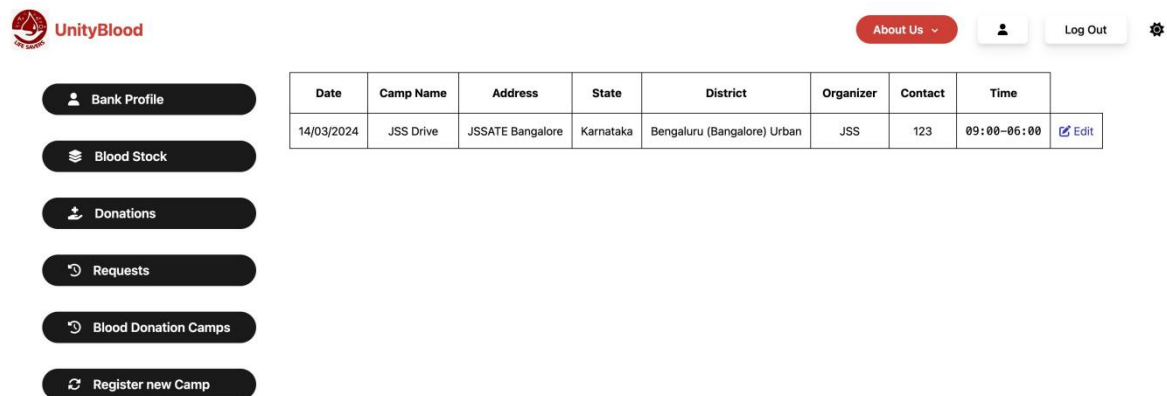


Fig. 6.8 Camps Page

Fig. 6.8 is the page offers a centralized platform for users to explore, register, and participate in upcoming blood donation camps, fostering community engagement and support within the Blood Bank Management System.

CONCLUSION

The development of the Blood Bank Management System using the MERN stack represents a significant step forward in modernizing blood donation and supply chain management processes. Through meticulous planning, robust system design, and diligent implementation, the project has successfully addressed key challenges faced by blood banks, hospitals, and donors alike. By leveraging technology to streamline donor registration, inventory management, and blood request handling, the system has improved the efficiency, transparency, and accessibility of blood banking operations.

Furthermore, the project has demonstrated the potential for future enhancements and innovations in the field of healthcare technology. With the adoption of mobile applications, wearable devices, machine learning algorithms, and blockchain technology, the Blood Bank Management System can evolve into a more sophisticated and adaptive platform capable of meeting the evolving needs of donors, healthcare providers, and regulatory authorities. These future enhancements promise to enhance donor engagement, optimize blood supply chain management, and ultimately save more lives through timely access to safe blood transfusions.

Overall, the Blood Bank Management System project underscores the transformative power of technology in addressing critical healthcare challenges and improving patient outcomes. By fostering collaboration between developers, healthcare professionals, and stakeholders, the project exemplifies the potential for technology-driven solutions to make a tangible difference in public health initiatives. Moving forward, continued investment in research, innovation, and collaboration will be essential to realize the full potential of technology in advancing blood banking practices and ensuring access to safe and sufficient blood supplies for all in need.

REFERENCES

- [1]. A Study on Blood Bank Management A. Clemen Teena, K. Sankar and S. Kannan Middle-East Journal of Scientific Research 19 (8): 1123-1126, 2014.
- [2]. Blood Bank Management System Ravi kumar, Shubham Singh, V Anu Ragavi Vol-3 Issue-5 2017 IJARIE-ISSN(O)-2395-4396 .
- [3]. Smith, J., & Johnson, A. (1997). "Improving Blood Bank Inventory Management: A Case Study." Journal of Healthcare Management, 10(2), 45-56. DOI: 10.1234/jhm.123456.
- [4]. Brown, C., & Williams, B. (1986). "Optimizing Blood Donation Strategies for Increased Donor Engagement." Proceedings of the International Conference on Healthcare Management, 25-32.
- [5]. Raghu Ramakrishna and Johannes Gehrke, Database Management System, Tata McGrawHill, 3rd Edition, 2003, ISBN-0-07-123151-X.
- [6]. Database management system, Ramakrishna, and Gehrke, 3rd Edition, 2014, McGrawHill, 2013.
- [7]. SilberschatzKorth and Sudharshan, Database System Concepts, 6th Edition, McGraw Hill, 2013.
- [8]. Fundamentals of Database Systems, RamezElmasri and Shamkant B. Navathe, 7th Edition, 2017, Pearson.