A WEB BASED BLOOD BANK MANAGEMENT SYSTEM TO FACILITATE THE PROCESS OF BLOOD DONATION AND MANAGEMENT IN REAL-TIME

A PROJECT REPORT

Submitted by

P SANTHOSH	**(711120243043)**
G SRI MURUGAN	**(711120243047)**
P SUBASH	**(711120243048)**

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

JANSONS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600 025

APRIL/MAY 2023

BONAFIDE CERTIFICATE

Certified that this AD8612-Socially Relevant Project report on "**A WEB BASED BLOOD BANK MANAGEMENT SYSTEM TO FACILITATE THE PROCESS OF BLOOD DONATION AND MANAGEMENT IN REAL-TIME**" is the bonafide work of **"P.Santhosh (711120243043), G.Sri Murugan(711120243047), P.Subash(711120243048)"** who carried out the project work under my supervision.

…… ……

SIGNATURE	**SIGNATURE**
Mr. M. VIVEK M. E	**Mr. M. VIVEK M. E**
HEAD OF THE DEPARTMENT	**SUPERVISOR**
Head and Assistant Professor	Assistant professor
Department of Artificial Intelligence	Department of Artificial Intelligence
Jansons Institute of Technology	Jansons Institute of Technology
Coimbatore-641659	Coimbatore-641659

Submitted for the practical examination project work viva-voce held on …….

……. ……..

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We would like to express our sincere thanks to the honorable **Chairman Rtn. MPHF. Shri. T.S. NATARAJAN** and Vice Chairmen **Mr. T.N. KALAIMANI & Mr. T.N. THIRUKUMAR** for providing all the facilities to do the project in the college campus and we have unique pleasure in thanking our respected **Principal Dr. V. NAGARAJAN M.E., Ph.D** for their continuous encouragement to do this project.

We express our gratitude to **Mr. M. VIVEK M. E** Head and Assistant Professor, Department of Artificial Intelligence and Data Science for his excellent guidance and for providing necessary facilities to carry out the project.

We would like to thank our project Supervisor **Mr. M. VIVEK M. E** Head and Assistant Professor, Department of Artificial intelligence and Data Science for his constant support and motivation in the success of this work.

We heartily express our thanks to the Project Co-ordinator **Mr. M. VIVEK M. E** Department of Artificial intelligence and Data Science for his guidance and suggestions during this project work.

We extend our sincere thanks to all Technical and non – Technical staff Members of our department who helped us in all aspects throughout this project.

We also thank the **GOD ALMIGHTY** for giving us courage and all the needful to fulfill this project.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The web application project aims to develop a real-time requesting system for blood bank management. The system will be designed to streamline the blood bank management process and provide healthcare providers with quick and easy access to critical information, allowing them to request specific blood products required for their patients in real-time. The proposed system will have a user-friendly interface that allows healthcare providers to interact with the system, and check for blood product availability. It will also include a application server and database server that manages user requests, stores data about donors and blood products. Overall, the proposed real-time requesting system for blood bank management can help healthcare providers improve patient outcomes, reduce errors, enhance efficiency, improve inventory management, increase communication, and improve patient safety.

CHAPTER-1
INTRODUCTION

Blood transfusion is an essential part of modern healthcare, allowing patients to receive critical blood products that can save their lives. Blood banks play a vital role in the management and distribution of blood products, ensuring that they are safe, properly screened and tested, and readily available for patients in need. However, managing blood products can be a complex and time-consuming process that requires careful coordination between healthcare providers and blood banks.

In recent years, advances in technology have led to the development of real-time requesting systems for blood bank management. These systems offer many advantages over traditional paper-based or phone-based systems, including improved patient care, reduced errors, enhanced efficiency, better inventory management, and increased communication. A real-time requesting system allows healthcare providers to quickly and easily request specific blood products required for their patients in real-time, allowing for faster delivery of blood products and potentially saving lives.

The proposed web application project aims to develop a real-time requesting system for blood bank management that is user-friendly, efficient, and secure. The system will provide healthcare providers with quick and easy access to critical information, allowing them to request specific blood products required for their patients in real-time. The system will also include a application server and database server that manages user requests, stores data about donors and blood products.

BLOOD BANK MANAGEMENT IN EARLY DAYS

In the early days of blood bank management, the process was much simpler and less sophisticated than it is today. The first successful blood transfusion was performed in 1665 by a British physician named Richard Lower, who transfused blood from one dog to another. However, it wasn't until the early 20th century that blood transfusions became a common medical practice.

Early blood banks were often run by hospitals and relied on volunteer donors to provide blood. The blood was collected and stored in glass bottles, and was often used immediately or within a few days of collection.

One of the biggest challenges in early blood bank management was determining blood type and compatibility. The ABO blood typing system, which is still used today, was discovered in 1901 by Karl Landsteiner. This made it possible to identify the different blood types and determine which ones were compatible for transfusion.

In the early days, blood transfusions were often risky and sometimes even deadly. There was a high risk of infection, and the process of matching blood types and cross-matching took time and was not always accurate.

Over time, improvements were made in blood collection, testing, and storage. Refrigeration allowed blood to be stored for longer periods of time, and more sophisticated testing methods made it easier to determine blood type and compatibility. Today, blood bank management is a highly regulated process that involves extensive testing, screening, and monitoring to ensure the safety of both donors and recipients.

BLOOD BANK MANAGEMENT OVER YEARS

1665: First successful blood transfusion is performed by British physician Richard Lower between dogs.

1901: ABO blood typing system is discovered by Karl Landsteiner.

1914: First blood transfusion is performed in a human patient.

1930s: Blood banks begin to be established, relying on volunteer donors to provide blood.

1940s: Refrigeration techniques are introduced, allowing for longer-term storage of blood.

1943: Blood typing and cross-matching become standard practice for blood transfusions.

1950s: Plastic blood bags are introduced, replacing glass bottles for blood storage.

1960s: Automated blood collection machines are developed, making the blood donation process faster and more efficient.

1970s: The Rh blood typing system is discovered.

1980s: Leukoreduction technology is introduced, reducing the risk of transfusion reactions.

1990s: Digital tracking systems and barcoding technology are introduced, improving inventory management and reducing the risk of human error.

2000s: Pathogen inactivation technologies are developed, reducing the risk of infection transmission through blood transfusion.

RISKS AT EACH PROCESSION OF YEARS

Throughout the history of blood bank management, there have been various risks associated with each process. Here are some of the major risks associated with blood bank management at different points in time:

1900s-1930s: Blood transfusions were often risky in the early days, as there was little understanding of blood typing and compatibility. Blood was often transfused without any testing or cross-matching, leading to a high risk of transfusion reactions and even death.

1940s-1950s: The introduction of refrigeration techniques allowed for longer-term storage of blood, but there were still risks associated with contamination and infection. Blood bags and storage containers were not always sterile, leading to the risk of bacterial or viral contamination.

1960s-1980s: The introduction of automated blood collection machines and leukoreduction technology reduced the risk of contamination and transfusion reactions, but there was still a risk of human error in the blood typing and cross-matching process. Transfusion reactions due to mismatched blood types or other factors were still a risk.

1990s-2000s: The introduction of digital tracking systems and barcoding technology improved inventory management and reduced the risk of human error, but there were still concerns about the risk of infectious diseases such as HIV and hepatitis C being transmitted through blood transfusions.

Today: Blood bank management is a highly regulated and safe process, but there are still risks associated with blood transfusions. These risks include allergic reactions, transfusion reactions, and the transmission of infectious diseases. However, extensive testing and screening protocols have greatly reduced these risks, making blood transfusions a safe and life-saving medical procedure.

DEVELOPMENT OF EXISTING SYSTEM OVER YEARS

The development of existing blood bank management systems has been focused on improving efficiency, accessibility, and overall effectiveness in managing the critical process of blood donation, storage, and distribution. Over the years, several advancements have been made to enhance the functionality and capabilities of these systems. Here is an overview of the key developments:

1. Computerization and Automation:
Computerization and automation have been instrumental in transforming blood bank management systems. The advent of computer technology and automated processes has revolutionized the way blood banks operate. Computerization involves the use of computers and software applications to digitize and streamline various aspects of blood bank operations. Tasks that were previously performed manually, such as donor registration, record-keeping, and inventory management, are now automated, eliminating the need for labor-intensive and error-prone manual processes. Automation, on the other hand, refers to the use of machines and equipment to perform tasks automatically. In blood banks, automated analyzers are utilized for blood typing, cross-matching, and screening tests, improving accuracy, efficiency, and turnaround times. Storage devices with automated temperature control systems ensure optimal storage conditions for blood units, reducing the risk of spoilage and wastage. Computerization and automation enable blood banks to process large volumes of data quickly and accurately, resulting in improved efficiency, reduced errors, and enhanced productivity. They also support real-time data access, allowing authorized personnel to retrieve information instantaneously and make informed decisions promptly.

2. Centralized Databases:

Centralized databases play a crucial role in the development of blood bank management systems. These databases serve as a centralized repository for storing and managing essential information related to donors, blood inventory, and test results. By consolidating data into a single location, centralized databases enable easy access, retrieval, and sharing of information across multiple blood bank locations. This promotes efficient coordination and communication between different departments within the blood bank, facilitating streamlined processes and effective decision-making. With a centralized database, blood bank staff can quickly retrieve and update donor records, ensuring accurate and up-to-date information. This includes details such as contact information, medical history, and blood type, which are crucial for matching donors with recipients. In terms of blood inventory management, centralized databases provide real-time visibility into the quantity, type, and expiration dates of blood units. This enables staff to monitor and track inventory levels effectively, preventing shortages or wastage of valuable resources. Furthermore, centralized databases enhance the traceability of blood units throughout their lifecycle. Each unit can be assigned a unique identifier, allowing for efficient tracking of its movement from collection to distribution. This traceability improves quality control, reduces the risk of errors, and ensures proper rotation and utilization of blood units. Centralized databases also support reporting and analytics capabilities. Data collected from various blood bank operations can be analyzed to derive insights into donor demographics, blood utilization patterns, and inventory management. These insights enable blood banks to make informed decisions, optimize resource allocation, and anticipate future demands.

3. Integration with Laboratory Equipment:

Integration with laboratory equipment has brought significant advancements to blood bank management systems. This integration allows for seamless communication and data exchange between the blood bank management system and various laboratory devices, enhancing efficiency, accuracy, and overall productivity. By integrating with laboratory equipment such as automated analyzers, blood bank management systems can directly receive and process test results. This eliminates the need for manual data entry, reduces the risk of transcription errors, and ensures real-time availability of accurate test information. The integration streamlines the testing process, shortens turnaround times, and enables faster decision-making regarding blood compatibility and suitability for transfusion. Furthermore, integration with storage devices, such as automated blood refrigerators, freezers, and platelet agitators, optimizes inventory management. These devices are equipped with sensors and monitoring capabilities that communicate with the blood bank management system. This integration enables real-time tracking of blood units, including their location, temperature, and expiry dates. It ensures that blood units are properly stored, rotated, and utilized, reducing the risk of waste and enhancing blood product safety. Integration with laboratory equipment also facilitates quality control and regulatory compliance. The blood bank management system can monitor and capture information on calibration, maintenance, and quality assurance processes performed on laboratory devices. This ensures that the equipment is functioning correctly and adheres to regulatory standards, ensuring the accuracy and reliability of test results. Moreover, integration enables the automatic transfer of information related to critical measurements and observations. For example, during blood typing and cross-matching, the results obtained from the automated analyzer are directly transmitted to the blood bank management system. This eliminates the need for manual data entry and minimizes the chances of human errors, improving the accuracy and integrity of patient and donor records. Integration with laboratory equipment also

supports advanced analytics and reporting capabilities. By consolidating data from various laboratory devices, the blood bank management system can generate comprehensive reports on blood utilization, test volumes, and quality control metrics. These insights help blood banks optimize resource allocation, identify trends, and make informed decisions regarding inventory management and process improvement.

4. Barcode and RFID Technology:

Barcode and RFID technology have revolutionized blood bank management systems by enabling efficient identification and tracking of blood units. Each unit is labeled with a unique barcode or RFID tag, which contains relevant information about the blood type, donor details, and expiration date. With barcode scanners or RFID readers, blood bank staff can quickly scan and retrieve detailed information about each unit, reducing the likelihood of errors and ensuring accurate identification. These technologies enhance traceability, improve inventory management, and facilitate the matching of blood units with patients, promoting patient safety and efficient utilization of blood resources.

5. Online Donor Registration and Appointment Scheduling:

Online donor registration and appointment scheduling have greatly transformed the blood donation process, making it more convenient and accessible for potential donors. With online platforms, individuals can easily register as blood donors by providing their personal information, medical history, and availability for donation. This eliminates the need for manual paperwork and allows blood banks to gather donor data efficiently. Appointment scheduling features enable donors to select a suitable date and time for their blood donation. This eliminates long waiting times and allows blood banks to manage donor flow effectively. By having a scheduled

appointment system, blood banks can ensure a consistent supply of donors, maintain a steady inventory, and optimize resource allocation. Online platforms also enable blood banks to communicate with registered donors effectively. They can send automated reminders and notifications about upcoming appointments, blood shortages, and special donation campaigns. This helps in engaging donors and fostering a sense of community involvement. Moreover, online donor registration and appointment scheduling platforms provide a centralized database of donors, making it easier for blood banks to access and manage donor information. This enhances the efficiency of donor recruitment, retention, and follow-up communications.

6. Mobile Applications:

Mobile applications have had a significant impact on blood bank management systems, offering a range of features and functionalities that enhance donor engagement, accessibility, and awareness. Mobile applications provide a convenient platform for potential donors to register, find nearby blood donation centers, and schedule appointments. These apps streamline the donor registration process, allowing users to input their information, medical history, and availability at their convenience. By making registration and appointment scheduling accessible on mobile devices, blood banks can reach a wider audience and encourage more individuals to participate in blood donation. Additionally, mobile apps enable blood banks to send notifications and alerts to registered donors. These notifications can include information about blood shortages, urgent donation requests, and upcoming donation campaigns. By leveraging push notifications, blood banks can efficiently communicate with donors, increasing donor retention and engagement. Mobile applications also play a crucial role in raising awareness about blood donation. They provide educational resources, news updates, and success stories related to blood

transfusions, encouraging individuals to become regular donors. Moreover, these apps can integrate with social media platforms, allowing users to share their donation experiences and spread the word about the importance of blood donation within their social networks. Furthermore, mobile apps can offer features such as gamification and loyalty programs to incentivize and reward donors. This can include badges, points, or special privileges for frequent donors, creating a sense of achievement and encouraging continued participation. Mobile applications also provide blood banks with valuable data and analytics. They can track donor demographics, donation patterns, and user engagement metrics. This data helps blood banks gain insights into donor behavior, optimize their outreach strategies, and make data-driven decisions to improve donation rates and overall operational efficiency.

7. Data Analytics and Reporting:

Data analytics and reporting have become crucial components of blood bank management systems, providing valuable insights and supporting informed decision-making. These capabilities leverage the vast amounts of data collected within blood banks to identify trends, optimize processes, and enhance operational efficiency. Data analytics tools enable blood banks to analyze donor demographics, donation patterns, and blood utilization metrics. By examining this data, blood banks can identify target populations for donor recruitment campaigns, develop strategies to increase donation rates, and forecast future blood demands. This helps in ensuring an adequate and steady supply of blood products. Furthermore, data analytics can help optimize inventory management. By analyzing historical usage patterns, expiration rates, and transfusion requirements, blood banks can accurately forecast demand, reduce wastage, and ensure appropriate stock levels. This ensures that blood units are available when needed while minimizing the risk of shortages or expired inventory. Data analytics also plays a vital role in ensuring blood product safety. By

analyzing test results, adverse events, and other quality control metrics, blood banks can identify potential risks or issues. This enables proactive measures to be taken to mitigate risks and maintain high standards of safety and quality in blood transfusion processes. Reporting capabilities within blood bank management systems allow for the generation of comprehensive reports and dashboards. These reports can include key performance indicators, operational metrics, and compliance statistics. They provide a holistic view of blood bank operations, enabling administrators and stakeholders to monitor performance, identify areas for improvement, and make data-driven decisions. Moreover, data analytics and reporting support regulatory compliance. Blood banks must adhere to strict regulations and standards to ensure patient safety and quality control. By analyzing data and generating compliance reports, blood banks can demonstrate their adherence to regulatory requirements and maintain their accreditation. Data visualization techniques, such as charts and graphs, make complex data easily understandable and accessible. This enables stakeholders to quickly grasp important trends, patterns, and performance indicators, facilitating effective communication and decision-making.

8. Interoperability and Integration: Blood bank management systems became more interconnected with other healthcare systems, such as electronic health records (EHRs) and hospital information systems (HIS). Interoperability facilitated seamless data exchange and integration, enabling blood banks to collaborate with healthcare facilities efficiently. Integration with EHRs and HIS improved patient safety by ensuring accurate and up-to-date information regarding blood transfusions.

9. Enhanced Safety Measures:
Enhanced safety measures have been paramount in blood bank management systems to ensure the safety of both donors and recipients throughout the blood donation and

transfusion processes. These measures aim to minimize the risk of infectious diseases, improve traceability, and maintain the integrity of blood products. One crucial safety measure is the implementation of stringent donor screening protocols. Donors undergo a thorough screening process to assess their eligibility and potential risk factors. This includes evaluating their medical history, travel history, and lifestyle factors that could impact the safety of donated blood. Donor screening helps identify individuals with infectious diseases or other conditions that could pose a risk to the recipient. Another vital safety measure is the implementation of robust testing procedures. Blood samples collected from donors are subjected to various laboratory tests to identify infectious diseases, such as HIV, hepatitis B and C, syphilis, and other transfusion-transmissible infections. Advanced automated analyzers are utilized to ensure accurate and efficient testing, minimizing the chances of false results. Furthermore, enhanced safety measures include the implementation of traceability systems. Each blood unit is assigned a unique identifier, such as a barcode or RFID tag, which allows for efficient tracking throughout its lifecycle. This traceability system enables the identification of the donor, the collection date, and relevant testing and processing information. It facilitates the recall of specific blood units in the event of any safety concerns or quality issues. The use of a comprehensive quality management system is another essential safety measure. Blood banks adhere to stringent quality control procedures to maintain the integrity and safety of blood products. This includes ensuring proper storage and transportation conditions, accurate labeling, and compliance with regulatory standards. Regular inspections and audits are conducted to assess compliance and identify areas for improvement. Safety measures also extend to transfusion practices. Proper identification protocols, including the verification of patient identity and blood compatibility, are implemented to ensure correct transfusions. Healthcare professionals receive training on safe transfusion practices and adhere to established

protocols to minimize the risk of errors during the transfusion process. Additionally, advanced technologies, such as pathogen inactivation systems, are being increasingly implemented to enhance safety measures. These systems use specific treatments, such as ultraviolet light or chemicals, to inactivate pathogens that may be present in donated blood. Pathogen inactivation technologies provide an additional layer of safety, reducing the risk of transfusion-transmitted infections.

10. Cloud-Based Solutions:

Cloud-based solutions have revolutionized the field of blood bank management by offering a range of benefits and capabilities. These solutions leverage cloud computing technology to provide scalable, secure, and cost-effective platforms for managing blood bank operations. One of the primary advantages of cloud-based solutions is their scalability. Blood banks can easily scale up or down their storage and computing resources based on their current needs. This flexibility allows them to handle fluctuations in blood supply and demand, ensuring optimal resource allocation without the need for significant infrastructure investments. Cloud-based solutions also provide enhanced accessibility. Authorized personnel can securely access the blood bank management system from anywhere with an internet connection, enabling remote management and collaboration. This accessibility is particularly valuable for blood banks with multiple locations or mobile collection units, as it facilitates real-time data updates and seamless coordination. Data backup and disaster recovery capabilities are inherent to cloud-based solutions. Blood bank data, including donor records, blood inventory, and test results, can be automatically backed up and stored in multiple geographically diverse locations. In the event of a system failure or disaster, data can be quickly restored, ensuring minimal disruption to blood bank operations and the preservation of critical information. Security is a top priority in cloud-based solutions. Cloud service providers implement robust

security measures, including encryption, access controls, and regular security updates, to protect data from unauthorized access or breaches. These measures often exceed the security measures that individual blood banks can implement on their own, providing an added layer of protection for sensitive information. Collaboration and integration are facilitated through cloud-based solutions. Blood banks can seamlessly integrate their cloud-based management systems with other healthcare systems, such as electronic health records (EHRs) or hospital information systems (HIS). This integration allows for efficient data exchange, ensuring accurate and up-to-date information regarding blood transfusions and patient care. Cost-effectiveness is another significant advantage of cloud-based solutions. Blood banks can avoid substantial upfront infrastructure costs and instead pay for the services they use on a subscription or pay-as-you-go basis. This pay-per-use model helps in reducing operational expenses and optimizing financial resources.

EFFECT OF INTERNET AND SOCIAL MEDIA OVER BLOOD BANK MANAGEMENT SYSTEM

1. Donor Recruitment: Internet and social media platforms provide powerful tools for donor recruitment. Blood banks can leverage these platforms to raise awareness about blood donation, share success stories, and encourage individuals to become donors. Through targeted advertising and online campaigns, blood banks can reach a larger audience and attract potential donors more effectively.

2. Online Donor Registration: The internet enables online donor registration, allowing individuals to easily sign up as blood donors through dedicated websites or mobile applications. This streamlined process eliminates the need for manual paperwork and enables blood banks to gather donor information efficiently. Social media platforms can also be used to promote online registration platforms and engage with potential donors.

3. Appointment Scheduling: Internet and mobile applications allow donors to schedule their donation appointments conveniently. They can select the date, time, and location that best suits them, reducing waiting times and optimizing donor flow. Blood banks can use online platforms to manage appointment slots and ensure a smooth donation process.

4. Communication and Engagement: Social media platforms provide blood banks with direct channels to communicate with donors. They can share updates on blood shortages, urgent donation needs, and upcoming campaigns. Blood banks can also use social media to express gratitude to donors, share donation success stories, and foster a sense of community and participation.

5. Donor Education: The internet serves as a valuable source of information for potential donors. Blood banks can utilize their websites and social media platforms to educate the public about blood donation, its importance, eligibility criteria, and the donation process. By providing accurate and accessible information, blood banks can address common misconceptions and encourage informed decision-making.

6. Donor Feedback and Reviews: Social media platforms allow donors to share their experiences and provide feedback on their donation process. Blood banks can monitor and respond to reviews and comments, addressing concerns and improving the donor experience based on valuable feedback.

1.1 BLOCK DIAGRAM

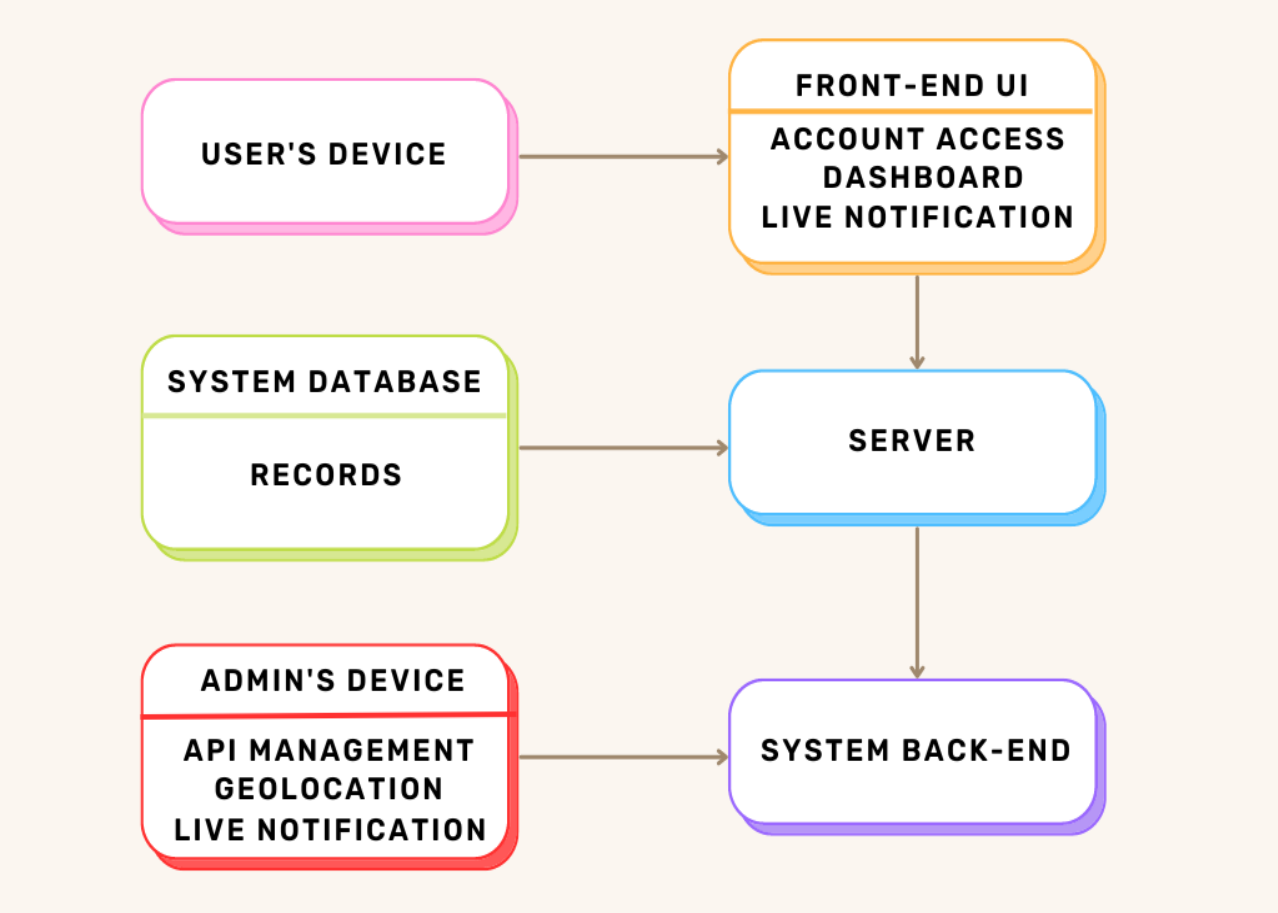

Fig 1.1 Block Diagram

1.2 MODULES DESCRIPTION

The project "**A WEB BASED BLOOD BANK MANAGEMENT SYSTEM TO FACILITATE THE PROCESS OF BLOOD DONATION AND MANAGEMENT IN REAL-TIME**" consists of modules such as,

- ➢ Registration Module:
- ➢ Login Module:
- ➢ Dashboard Module:
- ➢ Profile Management Module:
- ➢ Blood Request Module:
- ➢ Blood Donation Module:
- ➢ Notification Module:
- ➢ Blood Bank Management Module:
- ➢ Map Module

1.2.1 REGISTRATION

The registration module is an essential part of a real-time blood request web application that enables new users to create an account and access the system's features.

User Interface: The registration module should have a user-friendly interface that allows new users to fill in their details easily. The user interface should include fields for the user's name, email address, blood type, and other relevant information.

Validation: The registration module should include validation checks to ensure that the user's input is correct. For example, the email address should be in the correct format and not already in use in the system. The password should also meet the minimum requirements for length and complexity.

Verification: Once the user has filled in their details, the registration module should send a verification email to the user's Gmail address. The user will need to click on a link in the email to confirm their account and activate it. This step helps to prevent fake or fraudulent accounts from being created.

Error Handling: The registration module should include error handling to handle any errors that occur during the registration process. For example, if the user's email address is already in use, the system should display an error message asking the user to use a different email address.

Success Message: Once the user has completed the registration process and verified their email address, the registration module should display a success message. The

success message should provide the user with information on how to log in to the system and access its features.

User Account: Once the user has completed the registration process, the registration module should create a user account for the user in the database. The user account should include the user's name, email address, blood type, and other relevant information.

1.2.2 LOGIN

Login Module:

The login module is a crucial component of a real-time blood request web application that allows registered users to access the system's features.

User Interface: The login module should have a user-friendly interface that allows registered users to log in easily. The user interface should include fields for the user's Gmail address and password.

Validation: The login module should include validation checks to ensure that the user's input is correct. For example, the Gmail address should be in the correct format, and the password should match the one stored in the database.

Authentication: Once the user has filled in their Gmail address and password, the login module should authenticate the user's credentials. Authentication is a process of verifying the user's identity based on their login details. The login module should check the Gmail address and password against the records stored in the database to ensure that the user is valid.

Error Handling: The login module should include error handling to handle any errors that occur during the login process. For example, if the user enters an incorrect password, the system should display an error message asking the user to try again.

Session Management: Once the user's credentials have been verified, the login module should start a session for the user. A session is a way of identifying the user and maintaining their state within the system. The session should include information such as the user's name, email address, blood type, and other relevant information.

Redirect to Dashboard: Once the user has been authenticated and a session started, the login module should redirect the user to the appropriate dashboard based on their role (blood donor, recipient, or hospital staff).

Forgot Password: The login module should include a forgot password functionality to allow users to reset their password if they forget it. The user should be prompted to enter their email address, and a password reset link should be sent to their Gmail address. The password reset link should allow the user to enter a new password securely.

1.2.3 DASHBOARD

The dashboard module is an essential part of a real-time blood request web application that provides registered users with an overview of their account, relevant information, and actions they can take. Here's an elaboration of the dashboard module:

User Interface: The dashboard module should have a user-friendly interface that allows users to view their account information quickly.

Navigation: The dashboard module should include navigation elements that allow users to access different parts of the system quickly. For example, the navigation bar may include links to the user's profile, notifications, messages, and settings.

Requests: The dashboard module should display the user's current and previous blood requests. It should include details such as the date of the request, the requested blood type, the hospital's location, and the status of the request. Users should be able to view the details of their requests, including any updates or comments from the hospital staff.

Donations: If the user is a blood donor, the dashboard module should display their donation history. It should include details such as the date of the donation, the type of blood donated, and the location of the donation center.

Notifications: The dashboard module should display notifications for any new blood requests, updates to existing requests, and other important information.

Messaging: The dashboard module should include a messaging system that allows users to communicate with hospital staff and other users. The messaging system should be secure and allow users to send and receive messages in real-time.

1.2.4 PROFILE MANAGEMENT

The profile management module is an essential part of a real-time blood request web application that allows registered users to manage their profile information. Here's an elaboration of the profile management module:

User Interface: The profile management module should have a user-friendly interface that allows users to view and edit their profile information easily. The user interface should include fields for the user's name, email address, blood type, contact information, and other relevant details.

User Information: The profile management module should display the user's current profile information. Users should be able to view their name, email address, blood type, and other relevant details. If the user needs to update any information, they should be able to do so from the same screen.

Profile Picture: The profile management module should allow users to upload a profile picture. The picture should be displayed on the user's dashboard and other parts of the system where their profile information is visible.

Blood Type: The profile management module should allow users to update their blood type if necessary. Users should be able to select their blood type from a drop-down menu.

Contact Information: The profile management module should allow users to update their contact information, such as their phone number, address, and emergency contact details.

Password Change: The profile management module should allow users to change their password securely. The user should be prompted to enter their current password before entering the new password to ensure security.

Save Changes: The profile management module should include a save changes button that allows users to save any updates they make to their profile information.

1.2.5 BLOOD REQUEST

The blood request module is a critical part of a real-time blood request web application that allows hospitals or blood banks to request blood donations from registered users.

User Interface: The blood request module should have a user-friendly interface that allows hospitals or blood banks to create and manage blood requests easily. The user interface should include fields for the patient's name, blood type, hospital location, and other relevant details.

Blood Type: The blood request module should allow hospitals or blood banks to specify the blood type required for the patient. Users who have a compatible blood type will receive notifications about the request.

Location: The blood request module should allow hospitals or blood banks to specify the location of the patient. Users who are in the same or nearby areas will receive notifications about the request.

Urgency: The blood request module should allow hospitals or blood banks to indicate the urgency of the request. Users who are available to donate blood immediately will receive priority notifications.

Donor Selection: The blood request module should allow hospitals or blood banks to select the most suitable donor based on the blood type, location, and urgency of the request. Once a donor is selected, they will receive a notification about the request.

Donor Confirmation: The blood request module should require donors to confirm that they are available to donate blood before the request is approved. This step is

crucial to ensure that donors do not miss appointments or cancel appointments at the last minute.

Appointment Management: The blood request module should allow hospitals or blood banks to manage donor appointments efficiently. Hospitals or blood banks should be able to schedule appointments, send reminders to donors, and reschedule appointments if necessary.

Notification System: The blood request module should include a notification system that sends updates to donors and hospitals or blood banks about the status of the request. The notification system should also alert donors if the request is canceled or rescheduled.

Donor History: The blood request module should keep track of a donor's donation history. This feature allows hospitals or blood banks to identify donors who have donated blood in the past and have a history of reliable donations.

1.2.6 BLOOD DONATION

The blood donation module is a crucial part of a real-time blood request web application that allows registered users to donate blood.

User Interface: The blood donation module should have a user-friendly interface that allows users to view and manage their blood donation appointments easily. The user interface should include fields for the user's name, contact information, blood type, and other relevant details.

Blood Type: The blood donation module should allow users to specify their blood type. This feature is essential as it helps hospitals or blood banks find suitable donors quickly and efficiently.

Location: The blood donation module should allow users to specify their location. This feature helps hospitals or blood banks find donors who are close to their location.

Eligibility Criteria: The blood donation module should include eligibility criteria that determine whether a user is eligible to donate blood or not. The criteria should include factors such as age, weight, health conditions, and travel history.

Appointment Booking: The blood donation module should allow users to book blood donation appointments easily. Users should be able to select the date, time, and location of their appointment.

Appointment Confirmation: The blood donation module should require users to confirm their appointment before it is approved. This step is crucial to ensure that users do not miss appointments or cancel appointments at the last minute.

Donation History: The blood donation module should keep track of a user's donation history. This feature allows users to view their previous donations and helps hospitals or blood banks identify donors who have a history of reliable donations.

Notification System: The blood donation module should include a notification system that sends reminders to users about their upcoming appointments. The notification system should also alert users if the appointment is canceled or rescheduled.

1.2.7 NOTIFICATION

The notification module in a real-time blood request web application plays a critical role in keeping the users updated with the status of their blood requests and donation appointments.

1. Notification Settings: The notification module should allow users to set their notification preferences. Users should be able to choose the type of notifications they want to receive, such as email, SMS, or push notifications.

2. Blood Request Notifications: The notification module should send notifications to users when a hospital or blood bank accepts their blood request. It should also notify them when a suitable donor is found or when the request is fulfilled.

3. Appointment Reminders: The notification module should send reminders to users about their upcoming blood donation appointments. Users should be notified a day or two before their appointment to ensure that they don't forget or miss their appointment.

4. Appointment Cancellation Notifications: The notification module should send notifications to users if their blood donation appointment is canceled or rescheduled. This feature helps users plan their schedules accordingly and avoids any inconvenience.

5. System Alerts: The notification module should also send system alerts to the admin team in case of any technical glitches or system failures. The admin team can take immediate action to rectify the issue and prevent any inconvenience to the users.

6. Donation Confirmation Notifications: The notification module should send notifications to users once their donation is confirmed by the blood bank or hospital. This feature gives users a sense of satisfaction and encourages them to donate blood regularly.

7. Notification Log: The notification module should keep a log of all the notifications sent to the users. This feature helps the admin team monitor the system's performance and ensure that all users receive the necessary notifications.

1.2.8 BLOOD BANK MANAGEMENT

The blood bank management module in a real-time blood request web application is responsible for managing the inventory of the blood bank, tracking blood donation activities, and managing the blood supply chain.

1. Blood Inventory Management: The blood bank management module should allow blood banks to manage their blood inventory efficiently. Blood banks should be able to track the amount of blood they have in stock, the expiration dates of each unit, and the type of blood.

2. Donor Management: The blood bank management module should allow blood banks to manage their donors' data, including their blood type, contact information, donation history, and eligibility criteria.

3. Blood Request Management: The blood bank management module should allow hospitals and other organizations to submit blood requests to the blood bank. Blood banks should be able to approve or reject these requests based on their inventory levels and the availability of the requested blood type.

4. Blood Donation Management: The blood bank management module should allow blood banks to manage blood donation activities, including scheduling blood donation appointments, confirming donations, and maintaining donor records.

5. User Management: The blood bank management module should provide user management capabilities that allow blood banks to manage the access rights of their users. Blood banks should be able to add, delete, or modify user accounts based on their roles and responsibilities.

1.2.9 MAP

The map and location tracker module in a real-time blood request web application allows users to search for blood banks, hospitals, and donors within their vicinity. This module is essential in facilitating the process of blood donation and can save lives by enabling faster access to blood in emergencies.

1. Map Integration: The map and location tracker module should integrate with a mapping service, such as Google Maps or Mapbox, to provide users with an interactive map. The map should display the user's current location, as well as the locations of nearby blood banks, hospitals, and donors.

2. Location Tracker: The map and location tracker module should enable users to track the location of blood donors who have agreed to share their location. This feature can be useful in emergency situations where time is of the essence.

CHAPTER-2

LITERATURE SURVEY

Blood banks play a vital role in the healthcare system by ensuring the availability and efficient management of blood products. The effective functioning of blood banks is crucial for timely and safe transfusions, emergency situations, and surgeries. However, traditional blood bank management systems often rely on manual processes, paper-based records, and outdated inventory management methods, leading to inefficiencies and delays. In recent years, there has been a growing recognition of the need for real-time blood bank management systems to overcome these challenges and optimize blood bank operations. Real-time systems provide up-to-date information on blood inventory, streamline donor management processes, and enable seamless integration with hospitals for timely distribution. These systems utilize advanced technologies, such as barcode/RFID tracking, biometric identification, and cloud-based architectures, to improve efficiency, accuracy, and traceability in blood bank operations.

Traditional blood bank management systems have been the backbone of blood bank operations for many years. These systems typically rely on manual record-keeping, paper-based processes, and offline spreadsheets to manage donor information, track inventory, and facilitate blood product distribution. While they have served their purpose, traditional systems often suffer from various limitations and drawbacks that hinder their efficiency and effectiveness.One of the primary challenges of traditional blood bank management systems is the delay in information updates. With manual processes and paper-based records, it takes time to record and update donor information, blood inventory levels, and transaction details. This delay can lead to inaccuracies, redundancies, and difficulties in accessing real-time information, which

may have critical implications for patient care and emergency situations.Data redundancy is another common issue with traditional systems. Since these systems rely on manual data entry, there is a higher risk of duplication or inconsistent data across different records and documents. This redundancy not only consumes valuable time and effort in data verification but also increases the chances of errors and discrepancies in managing the blood inventory. Moreover, traditional blood bank management systems often lack real-time monitoring and reporting capabilities. The absence of automated tracking mechanisms for blood inventory and donor information makes it challenging to maintain accurate and up-to-date records. As a result, blood bank staff may face difficulties in identifying available blood products, monitoring expiration dates, and efficiently allocating blood units to hospitals in need.

The reliance on offline spreadsheets and paper-based processes also hampers the overall efficiency of traditional systems. The manual nature of these systems makes it time-consuming to search for specific donor information or retrieve historical data. Additionally, generating reports and analyzing data for decision-making purposes becomes a tedious task, often requiring extensive manual effort and data consolidation. Furthermore, traditional blood bank management systems lack integration capabilities with other healthcare systems. This lack of integration limits seamless communication and data exchange between blood banks and hospitals, hindering efficient blood product distribution and coordination during emergencies or urgent transfusion needs.

Real-time blood bank management systems are designed to overcome the limitations of traditional systems by utilizing advanced technologies and architectures. The architecture of a real-time blood bank management system consists of different

components that work together to ensure seamless donor management, inventory tracking, and efficient distribution of blood products. This section provides an elaborated overview of the architecture of a real-time blood bank management system.

The client-side application serves as the user interface, allowing blood bank staff to interact with the system. It provides functionalities for donor registration, blood product requests, inventory monitoring, and report generation. The client-side application can be accessed through various interfaces such as desktop applications, web-based interfaces, or mobile applications, offering flexibility and accessibility.

The database management system (DBMS) acts as the central repository for storing and managing all donor and blood product information. It stores data related to donor profiles, blood types, donation history, inventory levels, and transaction records. The DBMS ensures data integrity, security, and efficient retrieval for real-time processing and reporting.

The donor management component handles all aspects of donor information and engagement. It includes functionalities such as donor registration, eligibility screening, health history tracking, appointment scheduling, and donor communication. This component may incorporate biometric identification systems for accurate donor identification and verification.

The inventory management component is responsible for real-time monitoring and management of blood product inventory. It tracks the availability, storage conditions, and expiration dates of blood units. Technologies such as barcode or RFID can be

utilized for efficient tracking and identification of blood products. This component includes features for stock replenishment, automated alerts for low stock levels or expired units, and optimal allocation of blood units based on demand and compatibility.

The testing and quality assurance component handles blood testing processes, screening, and ensuring the safety and quality of blood products. It integrates with laboratory information management systems (LIMS) and automated testing equipment to streamline testing procedures and capture test results in real-time. This component includes features for result reporting, tracking test deviations, and maintaining comprehensive testing history for each blood unit.

The hospital integration component enables seamless integration between the blood bank management system and hospital information systems (HIS). It facilitates real-time communication and data exchange between blood banks and hospitals, ensuring efficient distribution of blood products based on specific hospital needs and emergencies. This component includes features for receiving and processing blood product requests, tracking deliveries, and providing timely updates to hospitals regarding available blood units and compatibility information.

The reporting and analytics component collects data from various components and generates comprehensive reports and analytics for decision-making. It enables blood bank administrators to analyze trends, track performance indicators, and make informed decisions regarding inventory management, donor recruitment, and process optimization. This component may include features for generating real-time dashboards, customizable reports, and data visualization tools.

Data security and access control mechanisms are integral to the architecture of a real-time blood bank management system. It includes features such as user authentication, role-based access control, encryption of sensitive data, and compliance with privacy regulations like HIPAA (Health Insurance Portability and Accountability Act). Strong security measures protect donor information, ensure confidentiality, and prevent unauthorized access or data breaches.

The architecture of a real-time blood bank management system may vary depending on the specific requirements, scalability needs, integration capabilities, and available infrastructure. It can be implemented using client-server architecture, cloud-based architecture, or a hybrid approach combining on-premises and cloud components. The choice of architecture depends on factors such as scalability, data security, cost-effectiveness, and infrastructure capabilities.

CHAPTER-3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Real-time blood bank management systems typically utilize a combination of various technologies to facilitate efficient and effective operations. Here are some existing technologies commonly employed in such systems:

1. Database Management Systems (DBMS): DBMS technology is crucial for storing, managing, and retrieving data related to blood donors, recipients, inventory, and transactions. It ensures data integrity, security, and enables quick access to real-time information.

2. Web-based Applications: Web-based applications provide a user-friendly interface for blood bank staff, donors, and recipients to interact with the system. These applications allow users to register, request or donate blood, search for donors, and track inventory in real-time.

3. Mobile Applications: Mobile apps extend the functionality of blood bank management systems to mobile devices. They enable donors and recipients to access the system, receive notifications, and perform tasks such as scheduling appointments, finding nearby blood banks, and requesting or donating blood.

4. Barcoding and RFID (Radio Frequency Identification): Barcoding and RFID technologies are used for tracking blood bags and inventory management. Each

blood bag is labeled with a unique barcode or RFID tag, allowing quick identification, tracking, and monitoring of blood units throughout the supply chain.

5. Integration with Laboratory Equipment: Blood bank management systems often integrate with laboratory equipment, such as blood analyzers and refrigeration systems. Integration enables seamless data exchange, automates test results entry, and ensures proper storage and handling of blood products.

6. Geolocation Services: Geolocation services, such as GPS (Global Positioning System), are utilized to locate nearby blood banks, hospitals, or donation centers. These services help donors and recipients find the nearest facilities for blood donation or transfusion.

7. SMS and Email Notifications: Blood bank management systems may incorporate SMS (Short Message Service) and email notifications to keep donors and recipients informed about appointment scheduling, blood availability, and other updates. This helps in maintaining communication and engagement with stakeholders.

8. Data Analytics and Reporting: Advanced analytics tools are employed to analyze blood bank data, identify trends, predict demand, and generate reports. These insights assist in optimizing inventory management, resource allocation, and decision-making processes.

DEMERITS IN EXISTING SYSTEM

While existing blood bank management systems offer numerous advantages, they may also have some demerits or limitations. Here are a few common drawbacks associated with such systems:

1. Lack of Interoperability: In some cases, blood bank management systems may lack interoperability with other healthcare systems or electronic health record (EHR) systems. This can hinder seamless data exchange and collaboration between different healthcare facilities, potentially leading to inefficiencies and delays in information sharing.
2. Limited Accessibility: Not all blood bank management systems may offer easy accessibility to all stakeholders, especially individuals with limited technological resources or those residing in remote areas with poor internet connectivity. This can restrict the participation of potential donors and recipients, limiting the effectiveness of the system.
3. Security Concerns: Blood bank management systems deal with sensitive and confidential data, such as personal health information and blood donor details. If the system lacks robust security measures, it may be vulnerable to data breaches, unauthorized access, or hacking attempts. This can compromise the privacy and confidentiality of the information stored within the system.
4. Reliance on Manual Data Entry: Some blood bank management systems may still heavily rely on manual data entry processes, which can be time-consuming and error-prone. Manual entry increases the risk of data inaccuracies, leading to potential problems in inventory management, donor matching, or recipient identification.
5. Limited Integration with Laboratory Equipment: While integration with laboratory equipment can streamline processes, some blood bank management systems may have limited compatibility or integration options. This may result in manual data

transfer from laboratory instruments to the system, leading to delays, potential errors, and increased administrative burden.

6. Maintenance and Upgrades: Depending on the specific system, maintenance and upgrades of blood bank management systems can be complex and require dedicated technical expertise. Lack of regular maintenance or delays in system updates can lead to performance issues, compatibility problems, and potential vulnerabilities.

7. Limited Real-Time Requesting Options: Sometimes unfortunate situations like lack of blood group we need may occur. In those circumstances we can only wait for nearby blood banks or hospitals to help us. Even those options may not be accessible in those situations so we need a more concrete system that may help us.

It's important to note that not all blood bank management systems will suffer from these demerits, and many organizations work to address these limitations through ongoing system improvements and advancements in technology.

3.3 PROPOSED SYSTEM

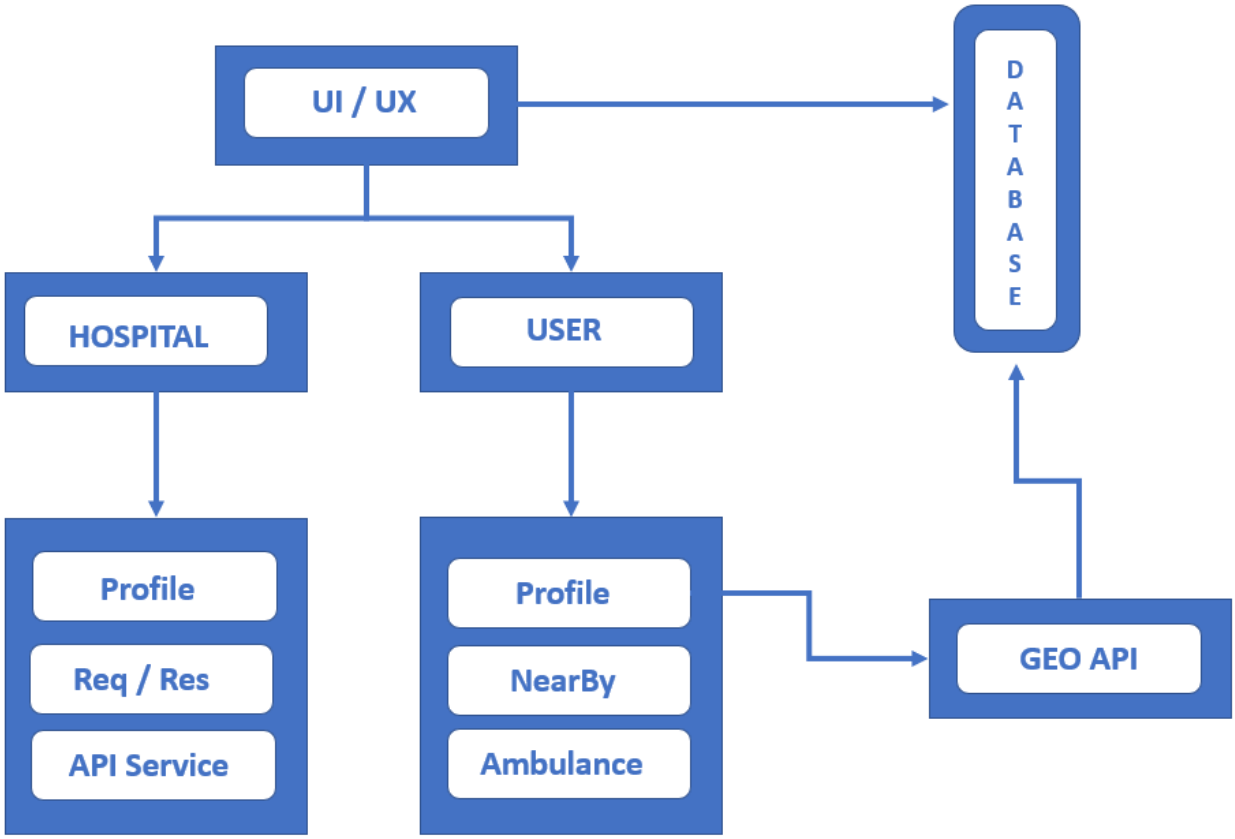

Fig 3.1 Proposed System Architecture

The proposed solution aims to develop a real-time blood donor requesting system that leverages geolocation services and contact list. The system will enable quick and efficient blood donor requests by identifying potential donors based on their proximity and utilizing the user's existing contacts. By combining these features, the proposed solution aims to streamline the process of finding suitable blood donors during emergencies or regular blood transfusion needs.

Features and Functionality:

1. User Registration and Profile Management:

- Users can register on the platform and create their profiles.
- Profiles will include essential details such as blood type, contact information, and consent to be a blood donor.

2. Geolocation Integration:

- The system will integrate geolocation services, such as GPS, to determine the user's current location.
- Users can search for nearby blood donors, enhancing the chances of finding donors who can quickly respond to the request.

3. Contact List :

- The system will securely integrate the contact list.
- Users can easily send blood donor requests to their contacts directly from the application.
- The system will automatically match the blood type requirement with the contacts who have registered as potential blood donors.

4. Real-Time Notifications:

- The system will send real-time notifications to registered blood donors in the vicinity when a blood donor request is initiated.
- Donors will receive notifications through push notifications.

5. Donor Response and Confirmation:

- Registered blood donors who receive requests will have the option to accept or decline based on their availability and willingness to donate.
- Once a donor confirms their availability, the system will notify the requester.

CHAPTER-4
SOFTWARE DESCRIPTION

4.1 SOFTWARE SPECIFICATION

- HTML
- CSS
- PHP
- Javascript
- SQL
- Bootstrap
- XAMPP
- Visual Studio Code

4.1.1 HTML

HTML, short for HyperText Markup Language, is the standard language used for creating and structuring web pages. It provides the foundation upon which websites are built. In this section, we will delve into HTML, exploring its definition, structure, elements, attributes, and the importance of using semantic HTML.

HTML serves as the backbone of a web page, allowing developers to define the structure and content of a website. It is a markup language that uses tags to enclose various elements and provide meaning to the browser. These tags are interpreted by web browsers to display the content in a structured manner.

The structure of an HTML document typically consists of three main sections: the doctype declaration, the head, and the body. The doctype declaration tells the browser which version of HTML is being used. The head section contains meta-information about the document, such as the title, character encoding, and links to external stylesheets or scripts. The body section contains the visible content of the web page, including text, images, links, and other multimedia elements.

HTML offers a wide range of elements that allow developers to define different types of content. Some of the commonly used elements include headings (h1 to h6), paragraphs (p), links (a), images (img), lists (ul, ol, li), tables (table, tr, td), forms (form, input, select), and many more. These elements provide structure and semantic meaning to the content, making it accessible and understandable for both humans and machines.

Attributes are additional properties that can be added to HTML elements to modify their behavior or appearance. For example, the "src" attribute in the "img" element specifies the URL of the image to be displayed, while the "href" attribute in the "a" element defines the URL to which the link should navigate. Attributes can also be used to specify styles, classes, IDs, and other metadata associated with an element.

One important aspect of HTML is semantic markup. Semantic HTML involves using elements that convey meaning and accurately describe the content they enclose. By using semantic elements like "header," "nav," "article," "section," "footer," and others, developers can provide more context and structure to their web pages. Semantic HTML improves accessibility, search engine optimization (SEO), and

overall code readability, making it easier for developers to understand and maintain the codebase.

In addition to the core HTML elements and attributes, HTML has evolved over time. The latest version, HTML5, introduced new elements and features that further enhanced the capabilities of web development. HTML5 introduced elements like "canvas" for drawing graphics, "video" and "audio" for embedding multimedia content, "article" and "section" for better structuring of content, and more.

HTML is a foundational technology in web development, and its importance cannot be overstated. It provides the structure and semantics necessary for web browsers to render web pages correctly. HTML is the starting point for creating accessible, SEO-friendly, and responsive websites. It is essential for developers to have a solid understanding of HTML and its best practices to create effective and well-structured web pages.

4.1.2 CSS

CSS, which stands for Cascading Style Sheets, is a style sheet language used to describe the visual presentation of a web page written in HTML. CSS enables web developers to control the layout, appearance, and design of web content, separate from its underlying structure. In this section, we will explore CSS in detail, covering its definition, selectors, properties, box model, layout, frameworks, and preprocessors.

CSS serves as the styling language for HTML documents, allowing developers to

modify the appearance of various elements, such as text, images, backgrounds, borders, and more. By separating the presentation layer from the content layer, CSS enhances the maintainability and flexibility of web pages.

One of the key concepts in CSS is selectors. Selectors define which HTML elements should be targeted and styled. CSS offers a wide range of selectors, including element selectors, class selectors, ID selectors, attribute selectors, pseudo-classes, and pseudo-elements. These selectors allow developers to target specific elements or groups of elements, applying styles to them selectively.

CSS properties determine the visual characteristics of selected elements. Properties control aspects such as color, font, size, spacing, alignment, and positioning. CSS offers an extensive list of properties, each affecting different aspects of element styling. Some commonly used properties include color, font-family, font-size, margin, padding, border, background, and many more.

The box model is a fundamental concept in CSS. It defines how elements are laid out and how their dimensions are calculated. According to the box model, every element is surrounded by four areas: content, padding, border, and margin. The content area contains the actual content, while padding adds space around the content. The border area surrounds the padding, and the margin area provides space between elements. Understanding the box model is essential for controlling spacing, alignment, and sizing of elements.

CSS plays a crucial role in defining the layout of web pages. It provides different techniques for positioning and arranging elements. Some common layout techniques

include normal flow, float, flexbox, and grid. Normal flow is the default layout behavior, where elements are stacked vertically one after another. Float allows elements to float to either the left or right, enabling text wrapping around them. Flexbox provides flexible box layout, allowing elements to be aligned and arranged in a flexible manner. Grid offers a powerful two-dimensional grid system for complex layouts.

To simplify and expedite CSS development, developers often utilize CSS frameworks. CSS frameworks, such as Bootstrap, Foundation, and Bulma, provide pre-built CSS styles and components. These frameworks offer ready-to-use styling and layout options, allowing developers to create responsive and visually appealing web pages quickly. CSS frameworks provide a consistent design language and save significant development time.

CSS preprocessors are another valuable tool in CSS development. Preprocessors like Sass (Syntactically Awesome Style Sheets) and Less add additional functionality to CSS, such as variables, mixins, functions, and nesting. These features make CSS code more modular, reusable, and maintainable. Preprocessors compile the code into regular CSS, which can then be used in web projects.

4.1.3 PHP

PHP, which stands for Hypertext Preprocessor, is a popular server-side scripting language used for web development. PHP enables the creation of dynamic and interactive web pages by embedding PHP code within HTML. In this section, we will explore PHP in detail, covering its definition, variables, data types, control

structures, functions, object-oriented programming, PHP frameworks, and its role in web applications.

PHP is widely used for server-side scripting, allowing developers to generate dynamic content on the server before sending it to the client's web browser. PHP code is executed on the server, and the resulting HTML is sent to the client for display. This server-side processing enables the creation of interactive web applications that can interact with databases, handle form submissions, and perform various other server-side tasks.

Variables are an essential component of any programming language, including PHP. PHP variables are used to store and manipulate data. In PHP, variables start with a dollar sign ($) followed by the variable name. PHP is a loosely typed language, meaning that variables do not require explicit data type declarations. Instead, the data type is determined based on the value assigned to the variable.

PHP supports various data types, including strings, integers, floats, booleans, arrays, and objects. Each data type has specific operations and functions associated with it. PHP provides functions for type conversion, manipulation, and validation of data.

Control structures in PHP allow developers to control the flow of execution based on certain conditions. PHP offers if-else statements, switch statements, loops (such as for, while, and foreach), and more. These control structures are used to make decisions, iterate over arrays, and perform repetitive tasks.

Functions in PHP allow developers to encapsulate a set of instructions into a reusable block of code. PHP has built-in functions for various tasks, such as string manipulation, date and time formatting, file handling, and database operations. Developers can also create their own functions to modularize code and improve code reusability.

PHP supports object-oriented programming (OOP) concepts, allowing developers to organize code into classes and objects. Classes define blueprints for objects, encapsulating properties (variables) and methods (functions) within a single unit. Objects are instances of classes that can interact with each other and manipulate data. OOP in PHP improves code organization, reusability, and maintainability.

PHP frameworks provide a structured approach to web application development, facilitating the creation of scalable and maintainable code. Popular PHP frameworks include Laravel, Symfony, CodeIgniter, and Zend Framework. These frameworks provide a wide range of features, such as routing, database abstraction, authentication, caching, and more. PHP frameworks follow the Model-View-Controller (MVC) architecture, separating the application logic, data, and presentation layers.

PHP is commonly used in web applications to interact with databases. PHP provides extensions and functions for connecting to various database systems, such as MySQL, PostgreSQL, and SQLite. This enables developers to store, retrieve, and manipulate data within web applications. PHP's database integration plays a crucial role in creating dynamic and data-driven web applications.

4.1.4 JAVASCRIPT

JavaScript is a widely-used programming language that is primarily used for developing interactive and dynamic web pages. It allows developers to add behavior, interactivity, and functionality to websites, making them more engaging and user-friendly. In this section, we will explore JavaScript in detail, covering its definition, variables, data types, operators, functions, object-oriented programming, frameworks, and libraries.

JavaScript is a client-side scripting language, meaning that it runs on the client's web browser rather than the server. When a web page is loaded, the JavaScript code embedded in the HTML is executed by the browser, enabling dynamic content and interactivity.

Variables are a fundamental concept in JavaScript, used to store and manipulate data. In JavaScript, variables are dynamically typed, meaning that they can hold values of different types. JavaScript supports various data types, including strings, numbers, booleans, arrays, objects, and more. Variables in JavaScript are declared using the "var," "let," or "const" keywords.

JavaScript provides a wide range of operators for performing mathematical, logical, and comparison operations. These operators include arithmetic operators (+, -, *, /), assignment operators (=, +=, -=, *=, /=), comparison operators (==, ===, !=, !==, >, <), logical operators (&&, ||, !), and more. Operators allow developers to manipulate and evaluate data in JavaScript.

Functions in JavaScript enable developers to encapsulate a set of instructions into a reusable block of code. JavaScript functions can be defined using the "function" keyword and can accept parameters and return values. Functions play a crucial role in code organization, modularity, and reusability.

JavaScript supports object-oriented programming (OOP) concepts, allowing developers to create objects and define their behavior through classes and prototypes. Objects are instances of classes or prototypes and can have properties (variables) and methods (functions). OOP in JavaScript promotes code reuse, encapsulation, and inheritance.

JavaScript frameworks and libraries provide additional functionality and abstraction to JavaScript development. Frameworks such as React, Angular, and Vue.js offer a structured approach to building web applications, providing features like component-based architecture, state management, routing, and more. Libraries, such as jQuery and Lodash, provide utility functions and simplified APIs for common tasks, enhancing development efficiency.

Asynchronous programming is an essential aspect of JavaScript. JavaScript uses an event-driven model, allowing developers to write non-blocking code that can handle multiple tasks concurrently. Asynchronous programming in JavaScript is achieved through techniques like callbacks, promises, and async/await, enabling tasks such as fetching data from a server, handling user interactions, and performing time-consuming operations without blocking the user interface.

JavaScript also has a wide range of built-in objects and APIs for accessing and manipulating various aspects of web pages and browser functionality. These include the Document Object Model (DOM) for manipulating HTML elements, the XMLHttpRequest object for making asynchronous HTTP requests, the localStorage and sessionStorage objects for client-side storage, and many more.

JavaScript is not limited to web development; it is also used in other environments like server-side development (Node.js) and mobile app development (React Native, Ionic). Its versatility and ubiquity make it one of the most popular programming languages globally.

4.1.5 SQL

SQL, which stands for Structured Query Language, is a standard language used for managing and manipulating relational databases. It provides a set of commands and syntax for creating, querying, modifying, and managing databases. In this section, we will explore SQL in detail, covering its definition, data manipulation, data definition, data control, joins, indexing, and its role in database management systems.

SQL is a declarative language that allows developers to interact with databases using simple and intuitive commands. It is designed to work with relational databases, which organize data into tables with rows and columns. SQL provides a uniform way to communicate with databases, regardless of the specific database management system (DBMS) being used, such as MySQL, Oracle, SQL Server, or PostgreSQL.

Data manipulation is a core aspect of SQL. It allows developers to retrieve, insert, update, and delete data from databases. The SELECT statement is used to retrieve data from one or more tables based on specified conditions. The INSERT statement is used to add new records to a table. The UPDATE statement is used to modify existing records. The DELETE statement is used to remove records from a table. SQL's data manipulation capabilities provide the means to interact with and manipulate data stored in databases effectively.

Data definition in SQL involves creating, modifying, and deleting database objects, such as tables, views, indexes, and constraints. The CREATE statement is used to create new tables, views, and other objects. The ALTER statement is used to modify the structure of existing objects. The DROP statement is used to delete tables, views, or other objects. With SQL's data definition capabilities, developers can define the structure of the database and its components.

Data control in SQL involves managing the permissions and access control for databases and database objects. SQL provides commands like GRANT and REVOKE to grant or revoke specific privileges to users or user roles. These privileges can include permissions to select, insert, update, delete, or execute certain SQL statements. Data control ensures the security and integrity of the database by limiting access to authorized individuals and controlling the operations they can perform.

Joins are a powerful feature in SQL that allows developers to combine data from multiple tables based on related columns. Joins enable complex queries that retrieve data from multiple tables in a single result set. SQL supports different types of joins,

including INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN, each serving a specific purpose in combining data.

Indexing is a technique used to optimize database performance by creating indexes on columns that are frequently used in queries. Indexes improve query performance by allowing the database to quickly locate the desired data without scanning the entire table. SQL provides commands like CREATE INDEX and DROP INDEX to create and remove indexes on tables. Proper indexing strategies can significantly enhance the speed and efficiency of database operations.

SQL plays a vital role in managing and interacting with databases. Database management systems provide an interface for executing SQL commands and managing the underlying data. SQL is used in a wide range of applications, from small-scale projects to enterprise-level systems. It is the backbone of data-driven applications, enabling efficient storage, retrieval, and manipulation of structured data.

SQL has evolved over time, and different versions of the language have been standardized. The most commonly used SQL standard is SQL:1999, also known as SQL-99. Subsequent versions, such as SQL:2003, SQL:2008, and SQL:2016, introduced new features and enhancements to the language. It is important for developers to be familiar with the specific SQL implementation and version supported by their chosen DBMS.

4.1.6 BOOTSTRAP

Bootstrap is a popular open-source front-end framework that provides a collection of pre-built HTML, CSS, and JavaScript components. It aims to simplify and expedite the process of developing responsive, mobile-first web pages and web applications. In this section, we will explore Bootstrap in detail, covering its definition, grid system, components, themes, customization, and its role in modern web development.

Bootstrap was developed by Twitter and was first released in 2011. It quickly gained popularity among web developers due to its extensive set of ready-to-use components and its focus on responsive design. Bootstrap is built on HTML, CSS, and JavaScript, making it compatible with different browsers and devices.

One of the key features of Bootstrap is its grid system. The grid system allows developers to create flexible and responsive layouts for their web pages. The grid system is based on a 12-column layout, which can be customized and adjusted to suit different design requirements. Developers can easily define the layout by assigning CSS classes to HTML elements, specifying the desired column widths for different screen sizes. The grid system simplifies the process of creating responsive designs that automatically adapt to different screen sizes, from desktops to mobile devices.

Bootstrap provides a wide range of pre-built components that developers can easily integrate into their web pages. These components include navigation bars, buttons, forms, cards, modals, carousels, dropdowns, and more. By using these components, developers can save significant development time and effort, as they don't have to

start from scratch to create common UI elements. The components are designed with a consistent style and behavior, ensuring a cohesive and professional-looking user interface.

Bootstrap also offers a variety of responsive utilities and classes that enable developers to control the visibility, alignment, spacing, and other aspects of their web page elements. These utilities allow developers to adjust the layout and appearance of their web pages based on different screen sizes, ensuring a seamless user experience across various devices.

Themes are another essential aspect of Bootstrap. Bootstrap provides a default theme, which includes a set of predefined CSS styles for typography, colors, buttons, and other elements. Developers can also choose to customize the theme by modifying the variables and CSS classes provided by Bootstrap. Alternatively, developers can opt for ready-made Bootstrap themes available from the community or commercial sources. Themes make it easy to achieve a visually appealing design without extensive CSS coding.

Customization is a significant advantage of Bootstrap. Developers can tailor Bootstrap to their specific needs by choosing which components, styles, and functionalities to include in their projects. Bootstrap provides a customization tool on its official website, where developers can select the desired components and customize various aspects such as colors, typography, and spacing. This allows developers to create a lean and optimized version of Bootstrap that meets their project requirements.

Bootstrap is also compatible with other JavaScript libraries and frameworks. It can be seamlessly integrated with popular JavaScript frameworks such as React, Angular, and Vue.js. This compatibility enhances the flexibility and versatility of Bootstrap, allowing developers to leverage its components and grid system within their preferred development environment.

Furthermore, Bootstrap has an active and supportive community. It offers extensive documentation, tutorials, and examples that help developers get started and learn the framework quickly. The community also contributes additional resources, themes, plugins, and extensions that further expand the capabilities of Bootstrap.

4.1.7 XAMPP

XAMPP is a popular open-source software package that provides a complete development environment for web developers. It stands for Cross-platform (X), Apache (A), MariaDB (M), PHP (P), and Perl (P), representing the core components of the package. In this section, we will explore XAMPP in detail, covering its definition, components, installation, configuration, and its role in local web development.

XAMPP is designed to simplify the setup and configuration of a local web server environment, allowing developers to create and test web applications on their own computer before deploying them to a live server. It provides a comprehensive solution that includes a web server (Apache), a database server (MariaDB), and a scripting language (PHP), along with additional tools like Perl and FileZilla FTP server.

The Apache web server is one of the most widely used web servers globally, known for its reliability and performance. XAMPP includes a pre-configured Apache server, making it easy to set up and start using. The Apache server enables the hosting of websites and provides support for various web technologies, including PHP, Perl, and CGI.

MariaDB is a popular open-source database management system that is fully compatible with MySQL. XAMPP includes a pre-configured MariaDB server, allowing developers to create and manage databases for their web applications. MariaDB provides a powerful and efficient database solution with support for complex queries, transactions, and data manipulation operations.

PHP is a server-side scripting language widely used for web development. XAMPP comes with a pre-installed and pre-configured version of PHP, allowing developers to execute PHP scripts and create dynamic web applications. PHP integrates seamlessly with the Apache server and MariaDB, enabling developers to build robust and interactive websites.

XAMPP also includes Perl, a versatile scripting language commonly used for web development, system administration, and network programming. Perl is known for its powerful text processing capabilities and its extensive library of modules. With Perl included in XAMPP, developers have the flexibility to choose Perl as their scripting language for certain projects.

Installing XAMPP is a straightforward process. It is available for multiple operating systems, including Windows, macOS, and Linux. The installation package provides a user-friendly installer that guides users through the installation process. Once installed, XAMPP can be started and stopped using a control panel or command-line interface, allowing developers to easily manage the web server and database server.

Configuration of XAMPP involves adjusting settings to suit specific development needs. The XAMPP control panel provides options to configure the Apache server, MariaDB, and other components. Developers can modify PHP settings, set up virtual hosts, manage database users and privileges, and configure other aspects of the environment. XAMPP also allows the installation of additional components and extensions, providing flexibility and customization options.

XAMPP is widely used by developers for various purposes. It provides a convenient environment for local web development, allowing developers to create and test websites without the need for an internet connection or a live server. XAMPP is particularly useful for prototyping, debugging, and learning web technologies. It provides a sandboxed environment where developers can experiment and refine their code before deploying it to a production server.

XAMPP also serves as a valuable tool for web application deployment in small-scale and personal projects. Once the web application is developed and tested locally, it can be easily migrated to a live server by transferring the necessary files and exporting the database. XAMPP's compatibility with other server environments ensures a smooth transition from the development environment to the production environment.

Additionally, XAMPP supports the development of content management systems (CMS) like WordPress, Joomla, and Drupal. These CMS platforms require a web server and database to run, which can be easily set up and managed with XAM

PP. Developers can install and configure CMS systems locally, customize themes and plugins, and then deploy the completed website to a live server.

4.1.8 VISUAL STUDIO CODE

Visual Studio Code (VS Code) is a lightweight and highly extensible source code editor developed by Microsoft. It has gained tremendous popularity among developers due to its powerful features, extensive customization options, and broad language support. In this section, we will explore Visual Studio Code in detail, covering its definition, features, extensions, customization, collaboration tools, and its role in modern software development.

Visual Studio Code is designed to provide a streamlined and efficient coding experience. It offers a clean and intuitive user interface that focuses on maximizing productivity. It supports a wide range of programming languages and provides features such as syntax highlighting, code completion, code formatting, and intelligent code navigation, which enhance the development process and help developers write clean and error-free code.

One of the key strengths of Visual Studio Code is its vast collection of extensions. Extensions are add-ons that enhance the functionality of the editor and allow

developers to customize their coding environment. VS Code has a rich extension marketplace where developers can find extensions for language support, debugging, version control integration, productivity tools, and more. These extensions enable developers to tailor their coding experience to their specific needs and work with their preferred technologies.

Visual Studio Code also excels in providing an integrated debugging experience. It supports debugging for a variety of programming languages and frameworks. Developers can set breakpoints, inspect variables, step through code, and troubleshoot issues directly from the editor. The debugging capabilities of VS Code greatly facilitate the process of identifying and fixing bugs, improving the overall software quality.

Another notable feature of Visual Studio Code is its strong integration with version control systems such as Git. It provides built-in Git support, allowing developers to manage source code repositories, track changes, commit, push, and pull code directly from the editor. This seamless integration streamlines the collaborative development process and makes it easy to work on projects as part of a team.

VS Code's customization options are extensive, enabling developers to personalize their coding environment to suit their preferences. It allows users to customize themes, fonts, color schemes, and keyboard shortcuts. Developers can also configure various settings related to indentation, auto-completion, linting, and formatting. With its rich customization capabilities, VS Code offers a highly tailored and comfortable coding experience.

Visual Studio Code is not only limited to local development but also supports remote development scenarios. It provides extensions and features for connecting to remote servers, containers, and virtual machines, enabling developers to work on projects hosted on remote environments without leaving the editor. This capability is particularly useful for collaborating with teams, working with cloud-based development environments, or accessing resources in different operating systems.

VS Code supports a wide range of productivity-enhancing features, such as integrated terminal, task automation, snippets, and intelligent code suggestions. These features help developers streamline their workflow, automate repetitive tasks, and write code more efficiently. The integrated terminal allows developers to run commands, build projects, and interact with the command-line interface without leaving the editor.

Visual Studio Code also offers rich language support, thanks to its extensive ecosystem and community contributions. It provides features specific to each programming language, such as language-specific extensions, linting, IntelliSense, and code formatting rules. The community actively contributes language-specific extensions, enabling developers to work with a wide range of programming languages and frameworks.

Collaboration is a significant aspect of modern software development, and Visual Studio Code provides tools to facilitate collaboration among developers. It offers features like Live Share, which allows multiple developers to work together in real-time, sharing their code, terminal, and debugging sessions. Live Share enhances collaboration, code reviews, and pair programming, enabling seamless teamwork, even for distributed teams.

CHAPTER-5
SYSTEM IMPLEMENTATION

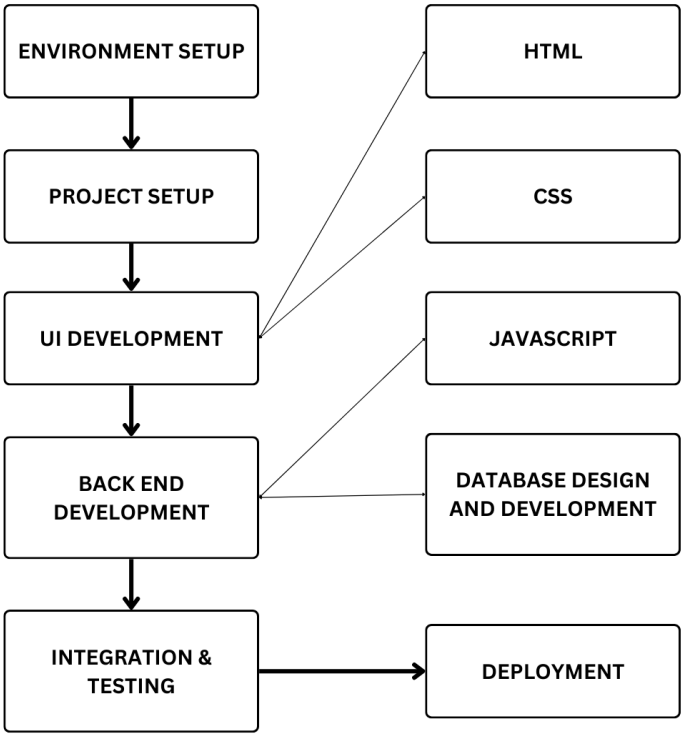

Fig 5.1 System Implementation Diagram

The system implementation phase is a crucial stage in the project lifecycle where the proposed solution is put into action. This phase involves the actual development, installation, configuration, and deployment of the system. In the case of this project, which aims to develop a web-based application using HTML, CSS, PHP, JavaScript, SQL, Bootstrap, XAMPP, and Visual Studio Code, the system implementation phase will involve several key steps. In this section, we will outline the system implementation plan.

1. Development Environment Setup:

The first step in the system implementation process is to set up the development environment. This involves installing the necessary software and tools required for the project. Install XAMPP, which includes Apache,MySQL, and PHP, to create a local web server environment. Additionally, install Visual Studio Code as the primary code editor. Configure XAMPP to ensure the web server and database server are running correctly.

2. Project Setup:

Create a new project folder on the local machine. Open Visual Studio Code and navigate to the project folder. Set up the basic file structure by creating HTML, CSS, PHP, and JavaScript files. Create additional folders for images, stylesheets, scripts, and other project assets. Initialize the project as a Git repository to enable version control and collaboration.

3. HTML and CSS Development:

Start by designing the user interface (UI) using HTML and CSS. Create the necessary HTML markup for different web pages, including headers, footers, navigation bars, content sections, and forms. Apply CSS styles to enhance the visual appearance and layout of the UI. Utilize Bootstrap's responsive grid system and pre-built CSS classes to achieve a consistent and mobile-friendly design.

4. PHP Development:

Integrate PHP into the project to add dynamic functionality to the web application. Develop PHP scripts to handle form submissions, user authentication, database interactions, and other server-side processes. Utilize PHP's built-in functions and libraries for common tasks, such as data validation and sanitization. Leverage PHP frameworks or libraries, if applicable, to expedite development and enhance code reusability.

5. JavaScript Development:

Enhance the interactivity and user experience of the web application using JavaScript. Develop JavaScript code to handle client-side form validation, perform AJAX requests, implement dynamic UI updates, and handle user interactions. Utilize JavaScript libraries or frameworks, such as jQuery or React, if required, to simplify development and leverage pre-built components.

6. Database Design and Development:

Design the database schema based on the application requirements. Use SQL to create the necessary tables, define relationships, and establish data constraints. Leverage the database management system MySQL provided by XAMPP to execute SQL queries and manage the database. Develop PHP scripts to interact with the database, including inserting, updating, retrieving, and deleting data.

7. Integration and Testing:

Integrate different components of the system, including HTML, CSS, PHP, JavaScript, and the database. Perform thorough testing to ensure the proper functioning of each feature and functionality. Test the application under various scenarios, including positive and negative test cases, to identify and resolve any bugs, errors, or usability issues.

8. Deployment:

Prepare the application for deployment to a production environment. Configure the web server (Apache) and database server MySQL for deployment. Optimize the application for performance and security. Transfer all project files to the production server, ensuring proper file permissions and directory structure.

CHAPTER-6

CONCLUSION

The proposed real-time blood donor requesting system, incorporating geolocation services and contact integration, aims to provide an efficient and effective solution for finding blood donors during emergencies. By leveraging the power of technology, this system can significantly enhance the chances of finding suitable donors quickly, potentially saving lives. It is essential to further refine and develop this proposed solution, taking into account the specific needs and requirements of blood banks, healthcare facilities, and potential users.

CHAPTER-7

FUTURE ENHANCEMENT

In the future, real-time blood bank management systems are likely to incorporate several advancements and developments to enhance their efficiency and effectiveness. Here are some potential future developments in real-time blood bank management systems:

Integration of Blockchain Technology: Blockchain technology can play a vital role in blood bank management systems by ensuring secure and transparent transactions, maintaining an immutable record of blood donations and transfusions, and facilitating efficient inventory management. Blockchain can enhance traceability and reduce the risk of errors or fraud in the blood supply chain.

Artificial Intelligence (AI) for Predictive Analysis: AI algorithms can analyze vast amounts of data from blood bank records, donor information, and patient needs to predict demand patterns and optimize inventory management. AI can also help in identifying potential donors, matching blood types, and reducing the risk of incompatible transfusions.

Enhanced Safety and Tracking Mechanisms: Future systems may integrate advanced safety measures, such as biometric identification for blood donors and recipients, to minimize the risk of errors and ensure accurate tracking throughout the blood supply chain. This can help in preventing transfusion-related complications and improving patient safety.

Interoperability: Blood bank management systems of the future are likely to focus on interoperability and seamless integration with other healthcare systems. This allows for efficient data exchange, improved coordination between different healthcare facilities, and streamlined workflows.

It's important to note that these are speculative developments and may or may not be implemented in real-world scenarios. However, advancements in technology continue to shape healthcare systems, and it is likely that blood bank management systems will incorporate innovations to improve their overall functionality and impact.

BIBLIOGRAPHY

REFERENCE

1. N. Adarsh, J. Arpitha, M. D. Ali, N. M. Charan and P. G. Mahendrakar, "Effective blood bank management based on RFID in real time systems," 2014 International Conference on Embedded Systems (ICES), Coimbatore, India, 2014, pp. 287-290, doi: 10.1109/EmbeddedSys.2014.6953176.
2. Real-Time Blood Donor Management Using Dashboards Based on Data Mining Models Sundaram, Shyam; Santhanam, T. International Journal of Computer Science Issues (IJCSI); Mahebourg Vol. 8, Iss. 5, (Sep 2011): 159-163
3. Aman Shah, Dev Shah, Devanshi Shah, Daksh Chordiya, Nishant Doshi, Rudresh Dwivedi,Blood Bank Management and Inventory Control Database Management System,Procedia Computer Science,Volume 198,2022,Pages 404-409,ISSN 1877-0509
4. E-Blood Bank Application For Organizing and Ordering The Blood Donation; Authors: Lilik Sumaryanti, Suwarjono Suwarjono, Lusia Lamalewa; Corresponding Author: Lilik Sumaryanti
5. P. A. J. Sandaruwan, U. D. L. Dolapihilla, D. W. N. R. Karunathilaka, W. A. D. T. L. Wijayaweera, W. H. Rankothge and N. D. U. Gamage, "Towards an Efficient and Secure Blood Bank Management System," 2020 IEEE 8th R10 Humanitarian Technology Conference (R10-HTC), Kuching, Malaysia, 2020, pp. 1-6, doi: 10.1109/R10-HTC49770.2020.9356980.
6. Ismail, O., Misra, S., Oluranti, J., Ahuja, R. (2021). Centralized Blood Bank Database and Management System. In: Singh, P.K., Singh, Y., Kolekar, M.H., Kar, A.K., Chhabra, J.K., Sen, A. (eds) Recent Innovations in Computing. ICRIC 2020. Lecture Notes in Electrical Engineering, vol 701. Springer.

APPENDIX

A. SCREENSHOTS

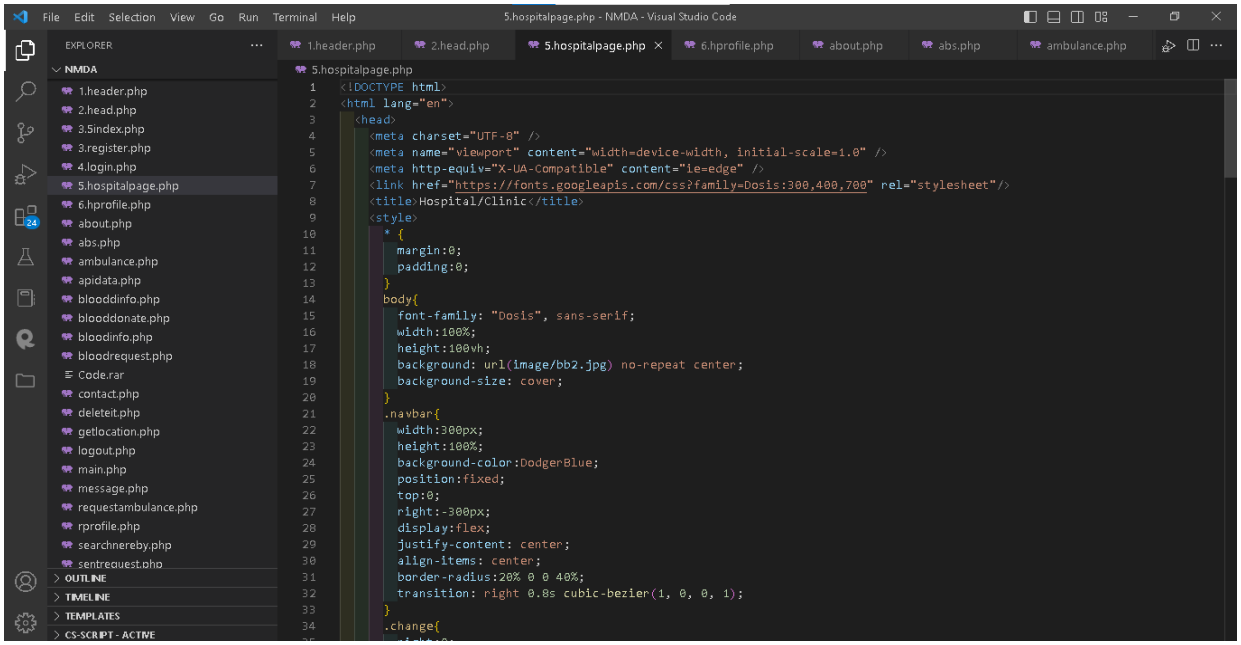

Fig 9.1 Sample Code Screenshot

B. SAMPLE SOURCE CODE

```
<nav class="navbar navbar-expand-sm navbar-light bg-light sticky-top">
   <div class="container">
      <a class="navbar-brand" href="index.php"><img src="image/favicon.jpg" width="30"
height="30" class="rounded-circle">Red Wave</a>

      <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#collapsibleNavbar"><i class="fas fa-align-left"></i></span>
      </button>

      <div class="collapse navbar-collapse" id="collapsibleNavbar">

     <?php if (isset($_SESSION['hid'])) { ?>

     <?php } elseif (isset($_SESSION['rid'])) { ?>

     <?php }  else { ?>
            <ul class="navbar-nav ml-auto">
       <li class="nav-item">
          <a class="nav-link btn btn-light" href="login.php">Login</a>
       </li>
       <li class="nav-item">
```

```
            <a class="nav-link btn btn-light" href="register.php">Register</a>
          </li>

        </ul>

        <?php } ?>
       </div>
    </div>
</nav>

<head>
    <title><?php echo $title ?></title>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <link                                                                rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>
    <link rel="stylesheet" type="text/css" href="../library/css/bootstrap.min.css"/>
    <script type="text/javascript" src="../library/js/jquery-3.2.1.min.js"></script>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"></script>
    <script src="https://kit.fontawesome.com/a076d05399.js"></script>
    <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'>
    <link href="css/style.css" rel="stylesheet"/>
    <link rel="shortcut icon" type="image/jpeg" href="image/favicon.jpg">
</head>
<?php
session_start();
if (isset($_SESSION['hid'])) {
  header("location:bloodrequest.php");
}elseif (isset($_SESSION['rid'])) {
  header("location:sentrequest.php");
}else{
?>
<!DOCTYPE html>
<html lang="en">
<head>
  <style>
body{
    background: url(jastimage/bb1.jpg) no-repeat center;
    background-size: cover;
    min-height: 0;
```

```
   height: 530px;
  }
.login-form{
   width: calc(100% - 20px);
   max-height: 650px;
   max-width: 450px;
   background-color: white;
}
</style>
</head>
<?php $title="Bloodbank | Register"; ?>
<?php require 'head.php'; ?>
<body onload="getlocation();">
  <?php include 'header.php'; ?>

   <div class="container cont">

   <?php require 'message.php'; ?>

     <div class="row justify-content-center">
       <div class="col-lg-4 col-md-5 col-sm-6 col-xs-7 mb-5">
         <div class="card rounded">
           <ul class="nav nav-tabs justify-content-center bg-light" style="padding: 20px">
             <li class="nav-item">
               <a class="nav-link active" data-toggle="tab" href="#hospitals">Hospitals</a>
             </li>
             <li class="nav-item">
               <a class="nav-link" data-toggle="tab" href="#receivers">User</a>
             </li>
           </ul>

   <div class="tab-content">

     <div class="tab-pane container active" id="hospitals">

      <form action="file/hospitalReg.php" method="post" enctype="multipart/form-data" class="myForm">
        <input type="text" name="hname" placeholder="Hospital Name" class="form-control mb-3" required>
        <input type="text" name="hcity" placeholder="Hospital City" class="form-control mb-3" required>
        <input type="tel" name="hphone" placeholder="Hospital Phone Number" class="form-control mb-3" required pattern="[0,6-9]{1}[0-9]{9,11}" title="Password must have start from 0,6,7,8 or 9 and must have 10 to 12 digit">
        <input type="email" name="hemail" placeholder="Hospital Email" class="form-control mb-3" required>
```

```
        <input type="password" name="hpassword" placeholder="Hospital Password" class="form-control mb-3" required minlength="6">
        <input type="hidden" name="hlatitude" class="form-control mb-3">
        <input type="hidden" name="hlongitude" class="form-control mb-3">
        <input type="submit" name="hregister" value="Register" class="btn btn-primary btn-block mb-4">
      </form>

    </div>


    <div class="tab-pane container fade" id="receivers">

      <form action="file/receiverReg.php" method="post" enctype="multipart/form-data" class="my_Form">
        <input type="text" name="rname" placeholder="User Name" class="form-control mb-3" required>
        <select name="rbg" class="form-control mb-3" required>
            <option disabled="" selected="">Blood Group</option>
            <option value="A+">A+</option>
            <option value="A-">A-</option>
            <option value="B+">B+</option>
            <option value="B-">B-</option>
            <option value="AB+">AB+</option>
            <option value="AB-">AB-</option>
            <option value="O+">O+</option>
            <option value="O-">O-</option>
        </select>
        <input type="text" name="rcity" placeholder="User City" class="form-control mb-3" required>
        <input type="tel" name="rphone" placeholder="User Phone Number" class="form-control mb-3" required pattern="[0,6-9]{1}[0-9]{9,11}" title="Password must have start from 0,6,7,8 or 9 and must have 10 to 12 digit">
        <input type="email" name="remail" placeholder="User Email" class="form-control mb-3" required>
        <input type="password" name="rpassword" placeholder="User Password" class="form-control mb-3" required minlength="6">
        <input type="hidden" name="rlatitude" class="form-control mb-3">
        <input type="hidden" name="rlongitude" class="form-control mb-3">
        <input type="submit" name="rregister" value="Register" class="btn btn-primary btn-block mb-4">
      </form>

    </div>
  </div>
  <a href="login.php" class="text-center mb-4" title="Click here">Already have account?</a>
```

```
</div>
</div>
</div>
</div>
<?php require 'footer.php' ?>
</body>
<script type="text/Javascript">
     function getlocation(){
       if(navigator.geolocation){
          navigator.geolocation.getCurrentPosition(showPosition,showError);
       }
   }
   function showPosition(position){
       document.querySelector('.myForm       input[name       =       "hlatitude"]').value       =
position.coords.latitude;
       document.querySelector('.myForm       input[name       =       "hlongitude"]').value       =
position.coords.longitude;
       document.querySelector('.my_Form       input[name       =       "rlatitude"]').value       =
position.coords.latitude;
       document.querySelector('.my_Form       input[name       =       "rlongitude"]').value       =
position.coords.longitude;
   }
   function showError(error){
     switch(error.code){
       case error.PERMISSION_DENIED:
       alert("You must Allow the Request for geolocation to fill out the form");
       location.reload();
       break;
     }
   }
</script>
</html>
<?php } ?>
<?php
session_start();
if (isset($_SESSION['hid'])) {
  header("location:bloodrequest.php");
}elseif (isset($_SESSION['rid'])) {
  header("location:sentrequest.php");
}else{
?>
<!DOCTYPE html>
<html>
<head>
  <style>
    body{
    background: url(image/RBC11.jpg) no-repeat center;
```

```
    background-size: cover;
    min-height: 0;
    height: 650px;
   }
.login-form{
    width: calc(100% - 20px);
    max-height: 650px;
    max-width: 450px;
    background-color: white;
}
</style>
</head>
<?php $title="Bloodbank | Login"; ?>
<?php require 'head.php'; ?>
<body onload="getlocation();">
  <?php require 'header.php'; ?>

    <div class="container cont">

      <?php require 'message.php'; ?>

      <div class="row justify-content-center">
        <div class="col-lg-4 col-md-5 col-sm-6 col-xs-7 mb-5">

          <div class="card rounded">
            <ul class="nav nav-tabs justify-content-center bg-light" style="padding: 20px;">
      <li class="nav-item">
        <a class="nav-link active" data-toggle="tab" href="#hospitals">Hospitals</a>
      </li>
     <li class="nav-item">
        <a class="nav-link" data-toggle="tab" href="#receivers">User</a>
     </li>
    </ul>

    <div class="tab-content">
       <div class="tab-pane container active" id="hospitals">
        <form action="file/hospitalLogin.php" class="login-form" method="post">
          <label class="text-muted font-weight-bold" class="text-muted font-weight-bold">Hospital Email</label>
          <input type="email" name="hemail" placeholder="Hospital Email" class="form-control mb-4">
          <label class="text-muted font-weight-bold" class="text-muted font-weight-bold">Hospital Password</label>
          <input type="password" name="hpassword" placeholder="Hospital Password" class="form-control mb-4">
          <input type="submit" name="hlogin" value="Login" class="btn btn-primary btn-block mb-4">
```

```
      </form>
     </div>


    <div class="tab-pane container fade" id="receivers">
      <form action="file/receiverLogin.php" class="login-form" method="post">
       <label class="text-muted font-weight-bold" class="text-muted font-weight-bold">User Email</label>
       <input type="email" name="remail" placeholder="User Email" class="form-control mb-4">
       <label class="text-muted font-weight-bold" class="text-muted font-weight-bold">User Password</label>
       <input type="password" name="rpassword" placeholder="User Password" class="form-control mb-4">
       <input type="hidden" name="latitude" class="form-control mb-3">
       <input type="hidden" name="longitude" class="form-control mb-3">
       <input type="submit" name="rlogin" value="Login" class="btn btn-primary btn-block mb-4">
      </form>
    </div>

  </div>
  <a href="register.php" class="text-center mb-4" title="Click here">Don't have account?</a>
</div>
</div>
</div>
</div>
<script>
  function getlocation(){
    if(navigator.geolocation){
      navigator.geolocation.getCurrentPosition(showPosition,showError);
    }
  }
  function showPosition(position){
    var latitude = document.querySelector('input[name = "latitude"]').value = position.coords.latitude;
    var longitude = document.querySelector('input[name = "longitude"]').value = position.coords.longitude;
    console.log(latitude);
    console.log(longitude);
  }
  function showError(error){
   switch(error.code){
    case error.PERMISSION_DENIED:
    alert("Please Allow Geolocation Service to Get The location");
    location.reload();
```

```
      break;
     }
    }
    </script>
<?php require 'footer.php' ?>
</body>
</html>
<?php } ?>
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <link                    href="https://fonts.googleapis.com/css?family=Dosis:300,400,700"
rel="stylesheet"/>
    <title>Hospital/Clinic</title>
    <style>
     * {
      margin:0;
      padding:0;
     }
     body{
      font-family: "Dosis", sans-serif;
      width:100%;
      height:100vh;
      background: url(image/bb2.jpg) no-repeat center;
      background-size: cover;
     }
     .navbar{
      width:300px;
      height:100%;
      background-color:DodgerBlue;
      position:fixed;
      top:0;
      right:-300px;
      display:flex;
      justify-content: center;
      align-items: center;
      border-radius:20% 0 0 40%;
      transition: right 0.8s cubic-bezier(1, 0, 0, 1);
     }
     .change{
      right:0;
     }

     .hamburger-menu{
```

```
  width:35px;
  height:30px;

  position:fixed;
  top:50px;
  right:50px;
  cursor:pointer;
  display:flex;
  flex-direction: column;
  justify-content: space-around;
}

.line{
  width:100%;
  height:3px;
  background-color:blue;
  transition: all 0.8s;

}

.change .line-1{
  transform:rotateZ(-405deg) translate(-8px, 6px);
}
.change .line-2{
  opacity:0;
}
 .change .line-3{
  transform:rotateZ(405deg) translate(-8px, -6px);
}

.nav-list{
  text-align:left;

}

.nav-item{
  list-style: none;
  margin:25px;
}

.nav-link{
  text-decoration: none;
  font-size: 22px;
  color: #eee;
  font-weight:500;
  letter-spacing: 1px;
  text-transform: uppercase;
```

```
      position:relative;
      padding:3px 0;
    }

    .nav-link::before,
    .nav-link::after{
      content: "";
      width:100%;
      height:2px;
      background-color: lightblue;
      position:absolute;
      right:0;
      transform:scalex(0);
      transition:transform 0.5s;
    }
    .nav-link::after{
      bottom:0;
      transform-origin:right;
    }
    .nav-link::before{
      top:0;
      transform-origin:left;
    }
    .nav-link:hover::before,
    .nav-link:hover::after{
      transform: scalex(1);
    }
  </style>

 </head>
 <body>
  <nav>
    <div class="container">
      <h1 style="border:5px solid blue; font-size: 35px; background-color:#d5f4e6; text-transform: uppercase; text-align: center; border-radius:15px; margin: 0px 500px 0px 500px; padding: 5px;">Hospital/Clinic</h1>
    </div>
  </nav>
    <div class="container">
    <nav class="navbar">
      <div class="hamburger-menu">
        <div class="line line-1"></div>
        <div class="line line-2"></div>
        <div class="line line-3"></div>
      </div>

      <ul class="nav-list">
```

```
            <li class="nav-item">
              <a href="hprofile.php" class="nav-link">My Account</a>
            </li>
             <li class="nav-item">
              <a href="bloodinfo.php" class="nav-link">Stock of Blood</a>
            </li>
             <li class="nav-item">
              <a href="searchnereby.php" class="nav-link">Search Nereby</a>
            </li>
             <li class="nav-item">
              <a href="bloodrequest.php" class="nav-link">Blood requests</a>
            </li>
             <li class="nav-item">
              <a href="deleteit.php" class="nav-link">Need blood</a>
            </li>
             <li class="nav-item">
              <a href="sentrequestd.php" class="nav-link">Status of your Blood request</a>
            </li>
            <li class="nav-item">
              <a href="requestambulance.php" class="nav-link">Ambulance Request</a>
            </li>
            <li class="nav-item">
              <a href="apidata.php" class="nav-link">API Service</a>
            </li>
            <li class="nav-item">
              <a href="logout.php" class="nav-link">LogOut</a>
            </li>
          </ul>
        </nav>
      </div>
      <script src="script.js"></script>
    </body>
  </html>
```

OUTPUT:

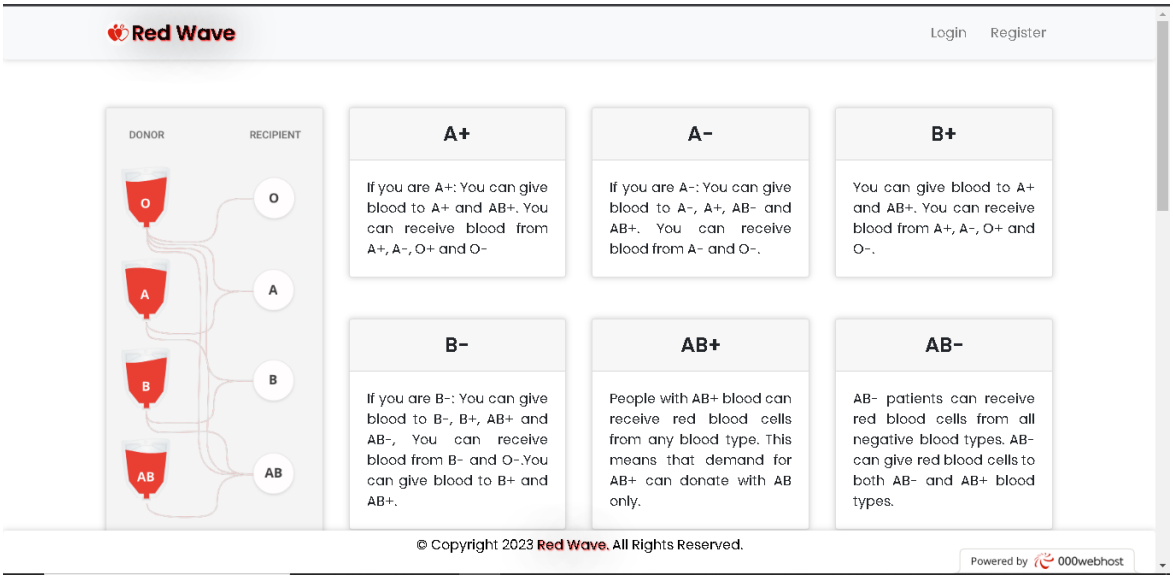

Fig 9.2 Project Output 1

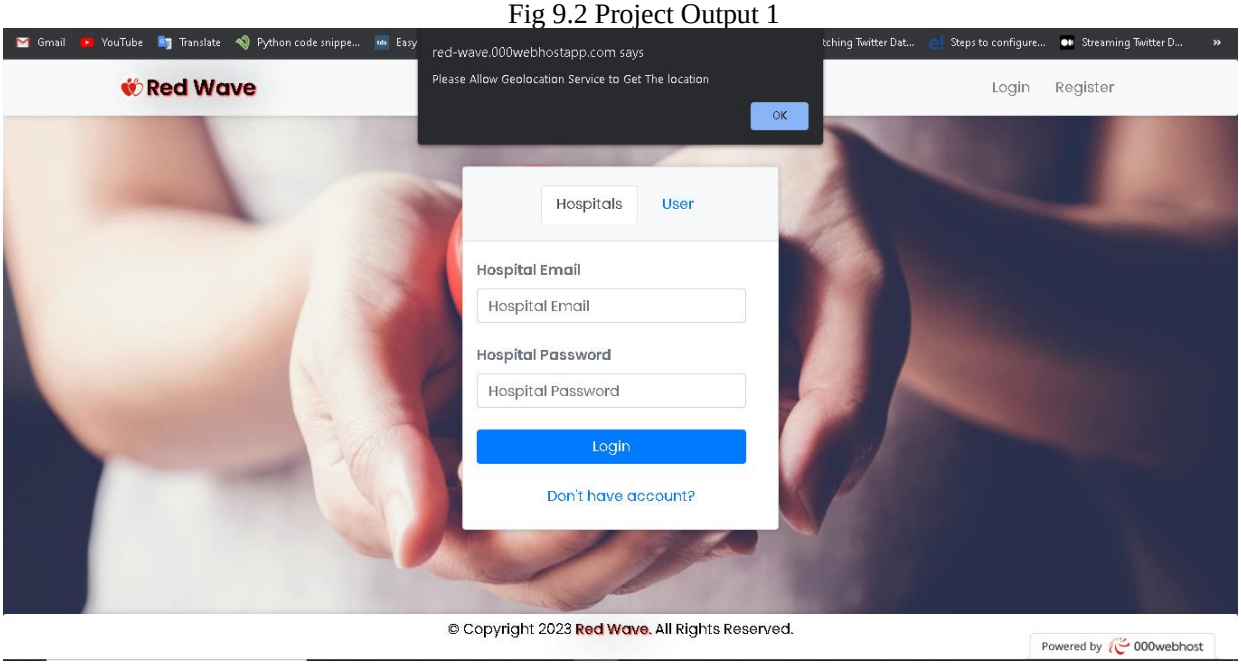

Fig 9.3 Project Output 2