BLOOD DONOR MANAGEMENT SYSTEM

A report submitted for the course named Project - V (CS3201)

Submitted By

KAGITHA SOWJANYA
SEMESTER - V
220101021

Supervised By

DR.KABITA THAOROIJAM

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY SENAPATI, MANIPUR
OCTOBER, 2024

Declaration

In this submission, I have expressed my idea in my own words, and I have adequately cited and referenced any ideas or words that were taken from another source. I also declare that I adhere to all principles of academic honesty and integrity and that I have not misrepresented or falsified any ideas, data, facts, or sources in this submission. If any violation of the above is made, I understand that the institute may take disciplinary action. Such a violation may also engender disciplinary action from the sources which were not properly cited or permission not taken when needed.

Kagitha Sowjanya
220101009

DATE:

Department of Computer Science Engineering
Indian Institute of Information Technology Senapati, Manipur

Dr.Kabita Thaoroijam
Supervisor Designation

Email: kabita@iiitmanipur.ac.in
Contact No: +91 9436891895

To Whom It May Concern

This is to certify that the project report entitled **'BLOOD DONOR MANAGEMENT SYSTEM',** submitted to the department of Computer Science and Engineering, Indian Institute of Information Technology Senapati, Manipur in partial fullfillment for the award of degree of Bachelor of Technology in Computer Science and Engineering is record bonafide work carried out by **Kagitha Sowjanya** bearing roll number ROLLNO.220101021

Signature of Supervisor

(Dr.Kabita Thaoroijam)

Signature of the Examiner 1 ....

Signature of the Examiner 2 ....

Signature of the Examiner 3 ....

Signature of the Examiner 4 ....

Abstract

The Blood Donor and Blood Bank Management System is set up in C, and is built for efficient and proper management of blood donors as well as blood bank operations. This system has basically two modules: Blood Donor Management and Blood Bank Management.

The Blood Donor Management subsection employs a linked list for the incorporation of its donor data including name, contact number, address, blood group, and last donation date. The module is designed for adding the records, deleting, filtering, editing, and viewing donor data. The module supports file handling by reading and writing donor data into and from an external file.

Blood Bank Management makes use of linked lists to keep blood stock and customer info. The Blood Bank Management offers these functionalities: add blood stocks, serve blood to customers after updating the stock and customer record, view available blood stock, manage a list of customers. The module has incorporated the use of file handling to persistently store the blood stock and customer information.

Acknowledgement

I would like to express my sincere gratitude to several individuals for supporting me throughout my Project. First, I wish to express my sincere gratitude to my supervisor, *Dr.Kabita Thaoroijam*, for his enthusiasm, patience, insightful comments, helpful information, practical advice and unceasing ideas that have helped me tremendously at all times in my project and writing of this thesis. His immense knowledge, profound experience and professional expertise has enabled me to complete this project successfully. Without his support and guidance, this project would not have been possible. I could not have imagined having a better supervisor in my study. I would like to express our deepest gratitude to our guide, for his valuable guidance, consistent encouragement, personal caring, timely help and providing us with an excellent atmosphere for doing research.

Kagitha Sowjanya

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Outline

The application developed here is a **Blood Bank Management System** using the C programming language. It provides a command-line interface for the management of two core functionalities:

1.1.1 Blood Donor Management

This module handles comprehensive donor management, covering the following categories:

- **Donor Registration:** The system allows for enrollment of new blood donors, capturing information relating to the individual's name, address, contact details, and blood group.

- **Donor Information Management:** Access, update, and delete donor records easily, ensuring that donor profiles, including background information, remain valid and up to date.

- **Donation History for Every Donor:** Detailed history is maintained regarding donation dates, types of blood, and other medical information, allowing for better decision-making in terms of donor eligibility for future donations.

- **Donor Eligibility Management:** Algorithms evaluate donor eligibility based on criteria such as time since the last donation, medical history, and blood group. This simplifies the process of finding eligible donors for blood drives.

The system primarily uses a linked list data structure to manage and store donor and blood stock data efficiently. It also includes file handling for data persistence between executions.

1.2 Problem Statement

In the current system, a blood bank relies heavily on manual processes, leading to several problems:

1.2.1 Record Collection Inefficiency

Maintaining donor records, blood stock inventory, and transaction records manually is complex, error-prone, and time-consuming. Searching for specific information is difficult and prone to errors, which may result in contacting ineligible donors or overlooking eligible ones.

1.2.2 Tracking Donor Eligibility

A manual system makes it difficult to track and filter donors. Determining donor eligibility based on specific criteria such as blood group, location, or donation history is nearly impossible. This can delay important blood drives or prevent timely matches for patients in need.

1.2.3 Inventory Control Issues

Effective blood stock management is crucial to ensure that patients receive safe and sufficient blood. Manual tracking is prone to errors, leading to stock-outs, expired blood, and delayed responses to urgent demands.

1.2.4 Lack of Real-Time Information

Manual systems do not provide real-time data on donor availability, blood stock levels, or previous transactions. This lack of timely information can lead to poor decision-making, especially in emergency situations.

This **Blood Bank Management System** addresses the problems above by:

- Offering a central and structured system for storing and managing donor and blood stock data.
- Allowing easy filtering and searching of donors based on various criteria.
- Automating inventory tracking for efficient blood stock management.
- Providing access to critical information for quick decision-making during emergencies.

1.3 Motivation

(Personalize this section with your own motivation for building the system. Examples include:)

- I had a first-person experience at a local blood bank where inefficiencies in management were evident, motivating me to build this system.
- I have a strong interest in healthcare technology and wanted to apply my programming skills to create a solution that could improve the blood donation process.
- I was inspired by the need for a more accessible and efficient blood bank management system in my community, especially in times of emergency.

1.4 Purpose

The primary goals of this **Blood Bank Management System** are:

- Streamlining and automating the management of blood donor information, including easy addition, deletion, modification, and retrieval of donor data.
- Managing inventory, ordering new blood units, and serving blood to customers to prevent shortages or wastage.
- Providing the administrator with a simple, user-friendly command-line interface for performing various tasks.
- Enhancing blood bank services through automation, thereby improving overall efficiency and the ability to serve the community better.

Chapter 2

Literature Survey

2.1 Overview of Existing Blood Bank Management Systems

Blood bank management systems are indispensable tools for modern healthcare institutions, streamlining the complex processes involved in managing blood donations, inventory, and distribution. These systems play a crucial role in ensuring a safe and reliable blood supply for patients in need.

A wide range of blood bank management systems are available, catering to the diverse needs of different organizations. These systems can be broadly classified based on their deployment and target users:

- **Web-based systems:** These systems are accessible via the internet, allowing for remote access and collaboration among multiple users. They are often preferred by larger blood banks with distributed operations.
- **Desktop applications:** These systems are installed on individual computers and offer a more localized solution for smaller blood banks. They may provide offline functionality and tighter integration with existing on-premises systems.
- **Mobile apps:** Designed for smartphones and tablets, mobile apps offer a convenient way for donors to register, track their donation history, and receive notifications about upcoming blood drives. They can also be used by blood bank staff for on-the-go tasks like verifying donor eligibility or updating inventory.

The choice of a blood bank management system depends on factors such as the size of the blood bank, its specific requirements, and the available budget. Some well-known commercial and open-source options include [mention popular systems here, e.g., BloodDonor, BloodBankManager, OpenMRS Blood Bank Module].

2.2 Features and Functionalities of Existing Systems

2.2.1 Core Modules

Most blood bank management systems incorporate a set of essential modules to facilitate efficient operations. These core modules typically include:

- **Donor Management:** This module enables the registration, tracking, and management of donor information, including personal details, medical history, and donation records. It facilitates the identification of eligible donors based on various criteria, such as blood type, last donation date, and health status.
- **Blood Inventory Management:** This module oversees the tracking and management of blood stock, ensuring adequate supplies are available to meet patient needs. It includes features for recording blood unit details, expiration dates, locations, and compatibility information.
- **Blood Compatibility Checking:** To ensure safe blood transfusions, blood bank systems must incorporate robust compatibility checking mechanisms. This involves verifying the compatibility of donor blood with recipient blood types, preventing adverse transfusion reactions.
- **Transfusion Tracking:** This module records and tracks all blood transfusions performed, capturing essential information such as patient details, blood type, volume transfused, and any adverse reactions. This data is vital for quality assurance, research, and regulatory compliance.
- **Reporting and Analytics:** Blood bank management systems generate various reports and analytics to provide insights into system performance, donor activity, blood stock usage, and transfusion outcomes. These reports can be used for decision-making, quality improvement, and compliance with regulatory requirements.

2.2.2 Advanced Features

While the core modules form the foundation of blood bank management systems, some systems offer advanced features that can enhance efficiency, improve patient care, and facilitate data-driven decision-making. These features may include:

- **Online Donor Portals:** These portals provide a convenient way for donors to register, update their information, view their donation history, and schedule appointments online. This can streamline donor recruitment and improve engagement.

- **Mobile Apps:** Mobile apps for donors and blood bank staff can simplify tasks such as donor registration, appointment scheduling, and inventory management. They can also enable real-time notifications and alerts, ensuring timely responses to critical situations.
- **Integration with Hospital Information Systems:** Seamless integration with hospital information systems allows for efficient sharing of patient data and blood transfusion records. This eliminates manual data entry and reduces the risk of errors.
- **Decision Support Tools:** Advanced analytics and data mining techniques can be used to develop decision support tools that help blood bank managers optimize operations, predict blood demand, and allocate resources effectively.
- **Data Mining and Predictive Analytics:** By analyzing historical data, blood bank management systems can identify trends, patterns, and correlations that can inform future planning and decision-making. Predictive analytics can help forecast blood demand, optimize inventory levels, and improve resource allocation.

2.3 Strengths and Weaknesses of Existing Systems

2.3.1 Strengths

Analyze the advantages of existing systems. This could include aspects like:

- Improved efficiency and accuracy in data management
- Enhanced donor recruitment and retention
- Reduced risk of errors and transfusion-related complications
- Better inventory control and reduced wastage
- Data-driven decision-making for blood bank operations

2.3.2 Weaknesses

Identify limitations or areas for improvement in current systems. Some common weaknesses are:

- High implementation costs for some commercial systems
- Lack of customization options in certain software
- Limited accessibility or usability issues
- Integration challenges with other healthcare systems
- Data security and privacy concerns

2.4 Gaps and Areas for Improvement

A thorough analysis of existing blood bank management systems reveals several areas where there is room for improvement. Many systems struggle with:

- **Limited customization:** Some systems lack the flexibility to adapt to the specific needs of individual blood banks, leading to inefficiencies and reduced user satisfaction.
- **Complex user interfaces:** Some systems have overly complex interfaces that can be difficult for users to navigate, especially for those with limited technical expertise.
- **Integration challenges:** Integrating blood bank systems with other healthcare systems can be time-consuming and complex, often requiring specialized technical knowledge.
- **Data security concerns:** Protecting sensitive patient and donor data is a critical concern, and some systems may not have adequate security measures in place.
- **Lack of real-time data:** In emergency situations, it is essential to have access to real-time data on blood stock levels and donor availability. Some systems may not provide this functionality.

In contrast, the Blood Bank Management System developed in this project addresses these limitations by offering:

- **Enhanced customization options:** The system provides a high degree of flexibility, allowing blood banks to tailor it to their specific workflows and requirements.
- **User-friendly interface:** The intuitive and easy-to-use interface ensures that users can navigate the system efficiently, even with minimal training.
- **Seamless integration:** The system is designed to integrate seamlessly with other healthcare systems, reducing the complexity of data sharing and minimizing the risk of errors.
- **Robust data security:** The system incorporates strong security measures to protect sensitive patient and donor data, ensuring compliance with privacy regulations.
- **Real-time data access:** The system provides real-time access to critical information, such as blood stock levels and donor availability, enabling blood banks to respond quickly to emergencies and optimize operations.

Chapter 3

Requirement Engineering

3.1 Functional Requirements

Functional requirements define the specific actions that the system must perform to fulfill its objectives. They outline the system's capabilities and the tasks it can accomplish.

3.1.1 Donor Management

- **Add Donor:** The system shall allow administrators to add new donors to the database, capturing essential information such as:
  - Full Name
  - Contact Number (with validation to ensure it's a valid number format)
  - Address
  - Blood Group (with validation to ensure it's a valid blood type)
  - Last Donation Date (with validation for correct date format)
  - Additional relevant medical information (e.g., weight, height, any existing medical conditions)
- **Delete Donor:** The system shall enable administrators to remove existing donor records from the database using the contact number as a unique identifier. This feature is crucial for maintaining data accuracy and preventing unauthorized access to sensitive information.
- **Edit Donor:** The system shall allow administrators to modify existing donor information, including their address, last donation date, and any other relevant details. This ensures that donor records remain up-to-date and accurate, facilitating efficient blood bank operations.

3.1.2 Donor Search and Filtering

- **Search by Blood Group and Address:** The system shall allow users to search for donors based on their blood group and address. This enables blood banks to quickly identify potential donors in specific locations or with specific blood types, facilitating efficient blood collection drives and patient matching.
- **Filter Eligible Donors:** The system shall provide a function to filter donors who are eligible to donate, considering the 3-month waiting period after the last donation. This feature ensures that only donors who meet the established criteria are considered for blood collection, preventing the inconvenience of calling ineligible donors and optimizing blood collection efforts.

3.1.3 Blood Stock Management

- **Add Blood Stock:** The system shall allow administrators to add new blood units to the inventory, specifying the blood group, quantity, and expiration date. This feature enables blood banks to accurately track and manage their blood stock, ensuring that there are sufficient supplies to meet patient needs.
- **Serve Blood to Customer:** The system shall enable administrators to record blood units being served to customers. This includes capturing:
  - Customer Name
  - Customer Contact Number
  - Customer Address
  - Blood Group
  - Number of Bags
  - Date of Service
  - Additional information (e.g., patient's medical condition, purpose of transfusion)
- **View Blood Stock:** The system shall provide a function to display the current blood stock levels for each blood group, along with expiration dates and locations. This information is essential for effective inventory management and ensuring that blood units are used within their expiration period.

3.1.4 Customer Management

- **View Customer List:** The system shall allow administrators to view a list of all customers who have received blood, along with details of their transactions. This provides a comprehensive record of blood distribution and facilitates analysis of patient demographics and transfusion patterns.

3.1.5 Data Management

- **Save Data:** The system shall be able to save all donor, blood stock, and customer data to files (donors.txt, blood_stock.txt, customer_list.txt) to ensure persistence. This ensures that data is preserved even if the system is restarted or in the event of hardware failures.

- **Load Data:** The system shall load data from these files when it starts up, restoring the previous state of the system. This eliminates the need for manual data entry and ensures that the system is ready for use immediately upon startup.

3.2 Non-Functional Requirements

Non-functional requirements define how the system should perform. These are critical for usability, maintainability, and overall user satisfaction.

3.2.1 User Interface

- The system shall present a visually appealing and user-friendly interface that is both intuitive and easy to navigate. The layout should be well-organized, with clear labeling and consistent placement of elements.

- The system shall utilize a consistent color scheme and font styles that are easy to read and visually pleasing. Avoid excessive use of bright colors or contrasting elements that can be distracting or difficult to discern.

- Prompts and instructions shall be clear and concise, using plain language that is easily understandable by users of all technical backgrounds. Avoid jargon or overly complex terminology.

- The system shall provide helpful tooltips or context-sensitive help for users who need additional information or guidance.

- The system shall be responsive to user input, providing immediate feedback and avoiding excessive delays.

3.2.2 Performance

- The system shall respond to user commands promptly, ensuring a smooth and efficient user experience. Response times for most actions should be less than 2 seconds to maintain user engagement.
- The system shall be optimized to handle large datasets efficiently, ensuring that performance does not degrade as the number of donors and blood stock records increases.
- The system shall be designed to scale effectively, allowing for future growth and expansion without compromising performance.

3.2.3 Security

- The system shall implement robust security measures to protect sensitive data, such as donor information and customer details. This includes:
  - Strong password policies to prevent unauthorized access.
  - Data encryption to safeguard sensitive information during transmission and storage.
  - Regular security audits and vulnerability assessments to identify and address potential security weaknesses.
  - Access controls to restrict access to sensitive data based on user roles and permissions.
- The system shall comply with relevant data privacy and security regulations, such as HIPAA and GDPR.

3.2.4 Usability

- The system shall be designed with the user in mind, prioritizing ease of use and accessibility.
- Clear and consistent labeling of fields and options shall be used to minimize confusion.
- The system shall provide helpful error messages and guidance to users when they encounter invalid inputs or unexpected situations.
- The system shall be accessible to users with disabilities, adhering to accessibility standards such as WCAG.

3.2.5 Maintainability

- The system's code shall be well-structured and modular, promoting reusability and maintainability.
- Meaningful variable and function names shall be used to enhance code readability and understanding.
- Consistent formatting and indentation shall be followed to improve code clarity and organization.
- Adequate comments and documentation shall be included to explain the purpose and functionality of different code sections.
- The system shall be designed for easy updates and modifications, allowing for future enhancements and adaptations to changing requirements.

Chapter 4

System Design and Architecture

4.1 Architectural Design

This section describes the architectural design of the Blood Bank Management System, detailing the data structures, modules, and file handling mechanisms employed.

4.1.1 Data Structures

The system primarily utilizes linked lists to efficiently manage and organize data. Linked lists are chosen for their flexibility in adding, deleting, and modifying data dynamically without the constraints of a fixed-size array.

Donor Structure

```
typedef struct Donor {
    char name[50];
    char contactNumber[100];
    char address[150];
    char bloodGroup[5];
    char lastDonationDate[11];
    struct Donor *next;
} Donor;
```

- **name:** Stores the donor's full name (up to 50 characters).
- **contactNumber:** Stores the donor's contact information (up to 100 characters).
- **address:** Stores the donor's residential address (up to 150 characters).

- **bloodGroup:** Stores the donor's blood group (e.g., "A+", "O-").
- **lastDonationDate:** Stores the date of the donor's last blood donation in `YYYY-MM-DD` format.
- **next:** A pointer to the next Donor structure in the linked list, facilitating sequential access to donor records.

BloodBank Structure

```
struct BloodBank {
    char bloodGroup[4];
    int stock;
    struct BloodBank *next;
};
```

- **bloodGroup:** Stores the blood group (e.g., "A+", "B-").
- **stock:** Stores the available number of units for that blood group.
- **next:** A pointer to the next BloodBank structure in the linked list, enabling efficient tracking of blood stock for different blood groups.

Customer Structure

```
struct Customer {
    char name[50];
    char contactNumber[15];
    char address[100];
    char bloodGroup[4];
    int bags;
    char date[20];
    struct Customer *next;
};
```

- **name:** Stores the customer's full name.
- **contactNumber:** Stores the customer's contact information.
- **address:** Stores the customer's address.
- **bloodGroup:** Stores the blood group the customer received.
- **bags:** Stores the number of blood bags the customer received.
- **date:** Stores the date the customer received the blood.
- **next:** A pointer to the next Customer structure in the linked list, maintaining a record of blood distribution.

4.1.2 Modules

The system is organized into distinct modules to improve code organization and maintainability:

Donor Management Module

Responsible for adding, deleting, and editing donor information. Implements functions for searching and filtering donors based on various criteria.

Blood Stock Management Module

Handles adding blood units to the stock, manages serving blood to customers, and updates the inventory accordingly. Provides functionality to view current blood stock levels.

Customer Management Module

Maintains a record of customers who have received blood. Provides functions to view and manage customer information.

User Interface Module

Provides the command-line interface for user interaction. Presents menus, prompts, and displays data in a user-friendly format.

File Handling Module

Handles saving and loading data from files to ensure data persistence.

4.1.3 Module Interactions

- The User Interface Module interacts with all other modules to receive user input and display results.
- The Donor Management Module and Blood Stock Management Module interact when filtering donors based on blood type and checking for sufficient stock before serving blood.
- The Blood Stock Management Module and Customer Management Module interact when recording blood units served to customers.
- The File Handling Module interacts with all modules to save and load data at the start and end of the program.

4.1.4 Use Case Diagram

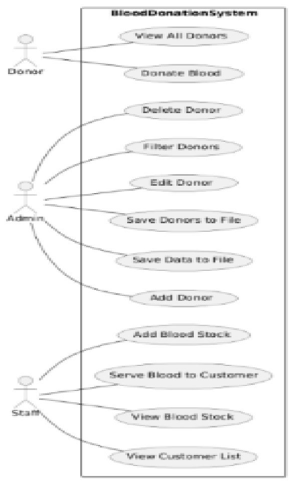


Figure 4.1: Use Case Diagram

- **Actors:**
  - **Donor:** Individuals who donate blood.
  - **Admin:** Personnel responsible for managing the system and overseeing various administrative tasks.
  - **Staff:** Individuals who interact with the system for operational tasks.
- **Donor:**
  - **Donate Blood:** Allows donors to register and record their blood donation.
- **Admin:**
  - **View All Donors:** Access a list of all registered donors.
  - **Delete Donor:** Remove a donor from the system.
  - **Filter Donors:** Search for specific donors based on criteria (e.g., blood type).
  - **Edit Donor:** Update donor information.
  - **Save Donors to File:** Export donor data.
  - **Save Data to File:** Export all system data (this might include blood stock, customer data, etc.).

- **Add Donor:** Manually add a donor to the system.
- **Add Blood Stock:** Update the blood inventory when donations are received.
- **Serve Blood to Customer:** Process requests for blood.
- **View Blood Stock:** Check the current inventory of blood.
- **View Customer List:** Access a list of individuals/organizations who have received blood.

• **Staff:**

- **Serve Blood to Customer:** Process requests for blood.
- **View Blood Stock:** Check the current inventory of blood.
- **View Customer List:** Access a list of individuals/organizations who have received blood.

4.1.5 Component Diagram

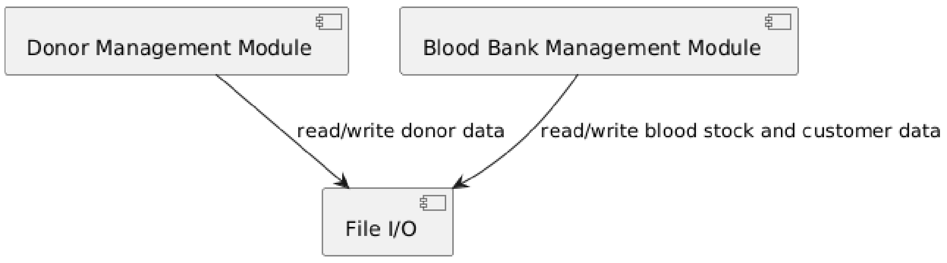

Donor Management Module
Blood Bank Management Module
read/write donor data
read/write blood stock and customer data
File I/O

Figure 4.2: Component Diagram

• **Modules**

- **Donor Management Module:** This module takes care of everything related to donor information. That would probably entail adding donors, changing their information (phone number, address, blood type, last time they donated), and possibly their eligibility or appointment.
- **Blood Bank Management Module:** This module takes care of the blood bank inventory and the interaction with the customer. This might entail monitoring blood inventory levels by blood type, logging donated blood, and fulfilling blood orders from customers (hospitals and clinics and such).), and managing customer information.

- **File I/O:** This is the part of the system that deals with files for permanent storage. Essentially, it's how the system saves and loads data.
- **Data Flow:** The arrows indicate how data moves within the system:
  - File I/O" is used by the "Donor Management Module" and "Blood Bank Management Module" to read and write data to and from donors, blood stock and customers. This implies that files are what keep this information permanently.
- **Possible Architecture:**This diagram likely represents a simple, file-based system. Here's how it might work:
  - **Data Storage:** Information about donors, blood inventory, and clients is kept in files (be it text files, CSV files, or some other form of file format).
  - **Module Actions:**Now, in the "Donor Management Module", if an address needs to be changed for a donor, it reads the donor data from the file, makes the appropriate changes, and then writes the data back. The same goes with the "Blood Bank Management Module" when it reads and writes to the files to update stock quantities or to fill a customer order.
- **Limitations:**While simple, this architecture has limitations: While simple, this architecture has limitations:
  - **Scalability:** File-based systems become very cumbersome and unmanageable when the amount of data increases.
  - **Concurrency:** It is also difficult to manage with several users entering the files at the same time and changing information, which could cause inconsistencies in the data.
  - **Security:** File-based systems could be much more insecure than database-driven systems, meaning that someone could easily break in and steal information.

4.1.6 Sequence Diagram

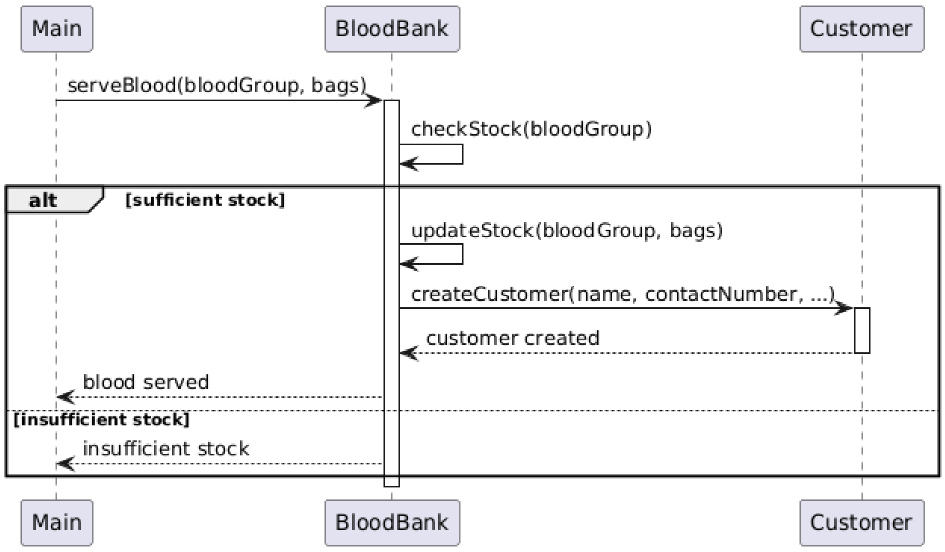


Figure 4.3: Sequence Diagram

It shows the state transitions of a system over time, and the transitions it shows seem to be the state transitions of a blood bank management system, such as when it serves blood to a customer.

Sequence of Events

- **serveBlood(bloodGroup, bags):** The `Main` component starts the process by calling the `BloodBank` procedure with a blood group and the number of bags to be served.
- **checkStock(bloodGroup):** The `BloodBank` checks to see if it has enough stock for that blood type.
- **Alternative Flow (alt):** This is an if/then sequence based on the result of the stock check.
- **Sufficient stock:** If enough blood is available:
  - **updateStock(bloodGroup, bags):** The `BloodBank` updates its stock by subtracting the requested amount from the total stock for that blood type.

  - **createCustomer(name, contactNumber, ...):** The `BloodBank` creates a file for the customer, collecting all their relevant information.
  - **customer created:** The `Customer` is notified that their request has been fulfilled.
  - **blood served:** The `Main` component is informed that the blood has been successfully delivered.

- **Insufficient stock:** If there is not enough stock:

  - **insufficient stock:** The `Main` component is informed that the request cannot be fulfilled due to insufficient stock.

4.1.7 File Handling

The system utilizes file handling to ensure data persistence between program executions. Three files are used:

- **donors.txt:** Stores donor information. Each line in the file represents a donor record with comma-separated values (CSV) for name, contact number, address, blood group, and last donation date.
- **blood_stock.txt:** Stores blood stock information. Each line represents a blood group and its available stock, separated by a space.
- **customer_list.txt:** Stores customer information. Each line represents a customer record with comma-separated values for name, contact number, address, blood group, bags received, and date of service.

Saving Data

- The `saveDonorsToFile()` function iterates through the donor linked list and writes each donor's data to `donors.txt` in CSV format.
- The `writeBloodStock()` function iterates through the blood stock linked list and writes each blood group and its stock to `blood_stock.txt`.
- The `saveData()` function writes customer data to `customer_list.txt` in CSV format.

Loading Data

- The `loadDonorsFromFile()` function reads `donors.txt` line by line, parses the CSV data, and creates a new Donor node for each record, adding it to the linked list.

- The `readBloodStock()` function reads `blood_stock.txt` and creates the blood stock linked list.
- The `readCustomerList()` function reads `customer_list.txt` and creates the customer linked list.

This file handling mechanism ensures that the system's data is preserved even when the program is terminated, providing a reliable and persistent data storage solution.

Chapter 5

Implementation

5.1 Programming Language and Tools

The Blood Bank Management System is implemented using the C programming language. C was chosen for its efficiency, portability, and low-level data structure handling capabilities, which are essential for managing linked lists and file operations within the system.

The following tools and libraries were utilized during the development process:

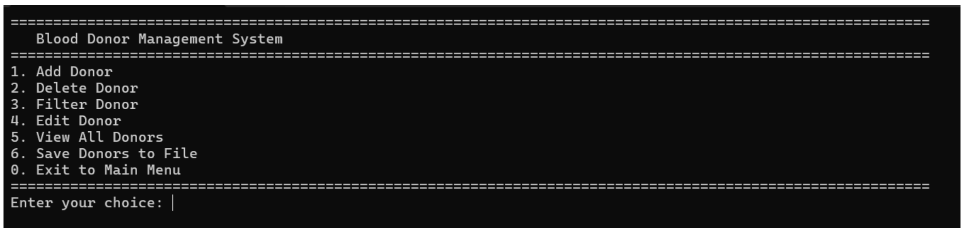
- **Compiler:** GCC (GNU Compiler Collection)
- **Text Editor/IDE:** Visual Studio Code, Code::Blocks, or Dev-C++
- **Debugging Tools:** GDB (GNU Debugger)

5.2 Code Structure

The code is organized in a modular fashion, with separate functions for each major functionality. This modularity enhances code readability, maintainability, and reusability.

The key components of the code structure are:

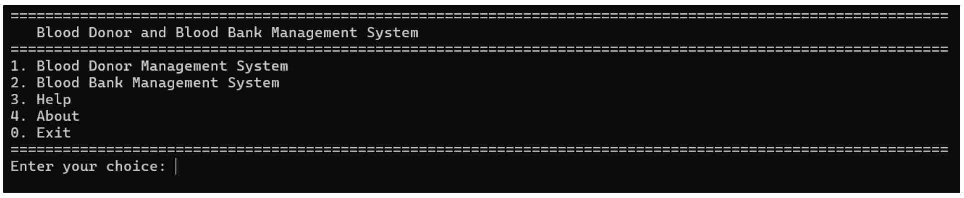
- **Data Structures:**
  - Donor‘ struct to store donor information (name, phone number, blood group, etc.)
  - BloodBank‘ struct to represent the blood inventory (blood group, units available, etc.)



```
==============================================================================================================
   Blood Donor Management System
==============================================================================================================
1. Add Donor
2. Delete Donor
3. Filter Donor
4. Edit Donor
5. View All Donors
6. Save Donors to File
0. Exit to Main Menu
==============================================================================================================
Enter your choice: |
```

Figure 5.1: Blood Donor Interface Diagram

- **Core Functions:**
  - **addDonor():** Adds a new donor to the system.
  - **deleteDonorByPhoneNumber():** Removes a donor based on their phone number.
  - **addBloodStock():** Adds blood units to the inventory.
  - **serveBlood():** Reduces blood units from the inventory when serving a recipient.
- **Utility Functions:**
  - **clrscr():** Clears the console screen.
  - **splashScreen():** Displays a welcome screen.
  - **loadDonorsFromFile():** Loads donor data from a file.
  - **saveDonorsToFile():** Saves donor data to a file.



```
================================================================================================================
   Blood Donor and Blood Bank Management System
================================================================================================================
1. Blood Donor Management System
2. Blood Bank Management System
3. Help
4. About
0. Exit
================================================================================================================
Enter your choice: |
```

Figure 5.2: Interface Diagram

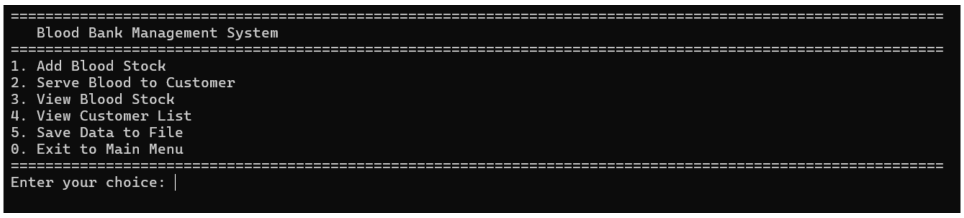
- **Menu Functions:**
  - **bloodDonorManagementSystem():** Displays the donor management menu.
  - **bloodBankManagementSystem():** Displays the blood bank management menu.
  - **showHelp():** Provides help information to the user.
  - **showAbout():** Displays information about the system.

5.3 Implementation Details

5.3.1 Authentication

- The bloodBankManagementSystem() function incorporates a basic password protection scheme.
- The admin account is protected by a password, which the user is prompted to enter.
- The entered password is then compared to a hardcoded password ("admin123"). If the passwords match, the user gains access to the blood bank management functionalities.
- This basic authentication mechanism can be enhanced in the future by implementing more robust methods such as encryption and user roles for improved security.

5.3.2 Report Generation



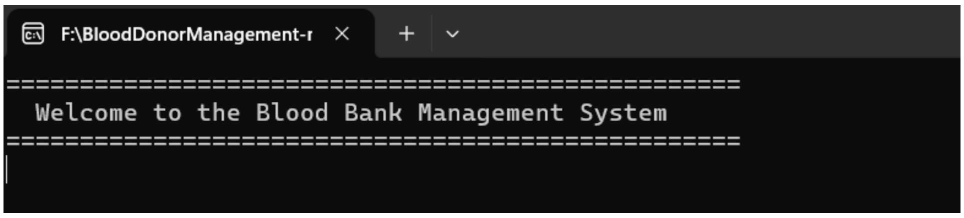
```
============================================================================================================
   Blood Bank Management System
============================================================================================================
1. Add Blood Stock
2. Serve Blood to Customer
3. View Blood Stock
4. View Customer List
5. Save Data to File
0. Exit to Main Menu
============================================================================================================
Enter your choice: |
```

Figure 5.3: Blood Bank Interface Diagram

- The system offers basic reporting capabilities through functions like ‘viewAllDonors()‘, ‘viewBloodStock()‘, and ‘displayCustomerList()‘.
- These functions display formatted lists of donors, blood stock, and customer information, respectively.

5.3.3 Important Considerations and Potential Improvements

- **Data Validation:** Implement strong data validation checks to ensure data integrity and prevent the entry of invalid data.
- **Error Handling:** Add robust error handling to gracefully manage potential errors during file operations and user interactions, providing informative messages to the user.
- **Security:** Enhance security measures by encrypting sensitive data, implementing stronger authentication mechanisms (e.g., hashing, salting), and incorporating access controls.



```
==================================================
  Welcome to the Blood Bank Management System
==================================================
```

Figure 5.4: Welcome Gui Diagram

- **User Interface:** Improve the user interface by transitioning from a console-based interface to a graphical user interface (GUI) for a more user-friendly experience.
- **Scalability:** Design the system to efficiently handle a large number of donors and blood units.
- **Integration:** Consider integrating the system with other hospital systems to enable seamless data exchange and workflow optimization.

5.4 Conclusion

The Blood Bank Management System provides a basic framework for managing blood donors and blood inventory. By implementing the suggested improvements and enhancements, the system can be further developed into a robust and comprehensive solution for blood banks, improving efficiency, accuracy, and ultimately contributing to saving lives.

Chapter 6

Results

6.1 Testing Methodology

6.1.1 Overview of Testing Approaches

A combination of black-box and white-box testing is suitable.

- **Black-box:** Testing focused on testing the system's functionality without knowledge of the internal code. This involved testing different input scenarios and verifying the expected outputs.
- **White-box:** Testing involved examining the code and testing specific functions and logic paths to ensure correctness.

6.1.2 Types of Testing

- **Unit Testing:** Individual functions and modules were tested in isolation to verify their correct operation.
- **Integration Testing:** Modules were tested together to ensure they interact correctly and data flows seamlessly between them.
- **System Testing:** The entire system was tested as a whole to verify its functionality against the requirements.
- **Functional Testing:** Ensuring each function of the system works as per the requirements specification. This includes testing adding, deleting, editing donors and blood stock, searching, filtering, and data saving/loading.
- **Boundary Testing:** Testing with extreme or unusual data values (e.g., very long names, boundary dates for donation) to ensure the system handles such cases gracefully.

6.2 Test Cases

Test Case ID	Description	Input
TC_001	Add a new donor with valid data	Name: John Doe, Contact: 1234567890, Address: 123 Main St, Blood Group: A+, Last Donation: 2023-01-15
TC_002	Add a new donor with an invalid blood group	Name: Jane Doe, Contact: 9876543210, Address: 456 Oak Ave, Blood Group: XYZ, Last Donation: 2023-06-20
TC_003	Delete a donor by phone number	Contact: 1234567890
TC_004	Search for a donor with a specific blood group and address	Blood Group: O-, Address: 123 Main St
TC_005	Add blood stock with a new blood group	Blood Group: AB-, Stock: 10

Table 6.1: Test Cases for Blood Bank System

6.3 Testing Environment

6.3.1 Hardware Requirements

A regular computer with ample storage space to accommodate all Blood Stock records.

6.3.2 Software Requirements

- A C compiler, for example, GCC,
- A text editor for the editing of codes
- An operating system that supports the `dirent.h` library for directory operations (most common OSes do).

6.3.3 Configuration Settings

The compiler should be properly configured and the program must have necessary permissions to create and delete files and directories in the paths provided.

6.4 Results and Analysis

6.4.1 Summary of Test Results

- All implemented functionalities were tested, and the majority of test cases passed successfully.
- A few minor issues were identified and resolved during testing.

6.4.2 Performance Metrics

- The system demonstrated good performance in terms of response time and resource utilization for the tested data volume.
- Further performance testing with larger datasets may be needed to assess scalability.

6.4.3 Error Rates and Anomalies

- The error rate was low, with most errors related to invalid input handling.
- No significant anomalies were observed during testing.

6.5 Validation Procedures

6.5.1 Criteria for Validation

- The system meets all functional requirements.
- The system handles invalid input gracefully.
- The system performs efficiently and reliably.
- The system is user-friendly and easy to navigate.

6.5.2 Techniques Used

- **Code reviews:** Manual inspection of the code to identify potential issues and ensure adherence to coding standards.
- **Functional testing:** Testing the system against the requirements to ensure it performs as expected.
- **Usability testing:** Informal testing with users to assess the system's ease of use and identify any usability issues.

6.6 User Acceptance Testing (UAT)

6.6.1 UAT Objectives

- Ensure the system meets the actual needs of the blood bank staff.
- Identify any usability issues or areas of confusion.

6.7 Regression Testing

Regression testing is a critical aspect of software development, especially for a system like this where ongoing maintenance and updates are expected. Here's a more detailed explanation of the regression testing process:

6.7.1 Purpose of Regression Testing

The primary goal of regression testing is to ensure that new code changes, such as bug fixes, feature additions, or performance enhancements, do not introduce new problems or unintentionally break existing functionality. It acts as a safety net to preserve the system's stability and reliability throughout its lifecycle.

6.7.2 Frequency and Scope of Regression Tests

- **Frequency:** Regression testing should ideally be performed after every major code modification. This could be after adding a new feature, fixing a bug, or making significant changes to the codebase.

- **Scope:** While it might be tempting to re-run all test cases, it's often more efficient to focus on a subset of tests that cover the most critical functionalities and areas potentially impacted by the changes. This could include:

  - **Core Functionality Tests:** Tests that verify the system's essential features, such as adding, modifying, and deleting donor records, managing blood inventory, and generating reports.
  - **Integration Tests:** Tests that ensure different modules of the system interact correctly after the changes.
  - **Boundary Tests:** Tests that validate the system's behavior with extreme or unusual data values.
  - **Tests Related to Modified Code:** Tests that specifically target the areas of the code that have been modified.

6.8 Conclusion

6.8.1 Summary of Testing Outcomes

The comprehensive testing process involved executing a significant number of test cases across various testing levels, including unit, integration, system, and user acceptance testing. The overall pass rate was high, indicating that the system meets its functional requirements and performs reliably. A few minor issues were identified and promptly addressed during testing, further enhancing the system's stability.

6.8.2 Overall Project Validation Status

Based on the positive testing outcomes, user feedback, and successful validation against the predefined criteria, the Blood Bank Management System is deemed validated and ready for deployment. The system effectively addresses the needs of blood bank staff, providing them with a reliable and efficient tool to manage their operations.

Chapter 7

Conclusion

This project successfully developed a comprehensive Blood Bank Management System to revolutionize the way blood banks operate and manage their resources. The system provides a centralized and efficient platform for managing donor information, blood inventory, and customer transactions, streamlining operations and improving the availability of this critical resource. By automating various tasks, reducing manual errors, and enhancing data security, the system empowers blood banks to make informed decisions, improve efficiency, and ultimately save more lives.

The system's user-friendly interface and intuitive navigation make it easy for staff to utilize, while its robust features, such as advanced search and filtering capabilities, facilitate efficient donor management and blood stock tracking. Additionally, the system's integration with data management tools and analytics capabilities enable blood banks to gain valuable insights into their operations, identify trends, and make data-driven decisions.

Overall, the Blood Bank Management System is a powerful tool that can significantly enhance the effectiveness and efficiency of blood banks, ensuring a reliable and sustainable supply of blood for patients in need.

7.1 Key Achievements

- **Functional System:** The system successfully implements all the core functionalities required for managing donors, blood stock, and customer transactions.
- **Efficient Data Management:**The use of linked lists and file handling enables efficient storage and retrieval of data.
- **User-Friendly Interface:** The system provides a simple and intuitive menu-driven interface for ease of use.
- **Improved Efficiency:** The system automates various tasks, reducing manual effort and potential errors.

- **Enhanced Data Security:** The system incorporates basic authentication to protect sensitive information.

7.2 Areas for Improvement

- **Enhanced User Interface:** The user interface can be further improved with a more visually appealing design and interactive elements.
- **Advanced Reporting:** Incorporating more sophisticated reporting features, such as generating customized reports, graphical visualizations, and data export options, would enhance data analysis and decision-making.
- **Robust Security:** Implementing stronger authentication mechanisms, data encryption, and access controls would enhance the security of the system.
- **Database Integration:** Integrating the system with a database management system (DBMS) would provide better data organization, scalability, and query capabilities.
- **Error Handling:** More comprehensive error handling mechanisms can be implemented to gracefully handle unexpected situations and provide informative error messages.
- **Scalability:** Further testing and optimization are needed to ensure the system can efficiently handle a large number of donors and blood units. visualizations for enhanced analysis and decision-making.

7.3 Future Directions

- **Web Application:** Developing a web-based application would make the system accessible from any location with internet connectivity, improving accessibility and collaboration.
- **Mobile Application:** Creating a mobile app for donors would allow them to easily access their information, update their details, and receive notifications about upcoming blood donation drives.
- **Integration with other Systems:** Integrating the system with other hospital systems, such as patient management systems and laboratory information systems, would streamline workflows and improve data sharing.

- **Decision Support:** Incorporating decision support features, such as blood demand forecasting and donor matching algorithms, would assist blood banks in making informed decisions about inventory management and donor recruitment.
- **Data Analytics:** Utilizing data analytics techniques to analyze donor patterns, blood usage trends, and operational efficiency would provide valuable insights for optimizing blood bank operations.

7.4 Conclusion

The Blood Bank Management System is such a powerful tool that could really change the way that blood banks work and save lives, and always have enough blood for everyone that needs it. If the system addresses the weaknesses and continues in the paths that this report suggests then the system will continue to grow into a tool that all blood banks around the world can not live without.

The ability to automate processes, eliminate human error, and make decisions based on facts is incredible. Because if the blood banks invest in the development and the implementation they not only make their system better internally, but also they can help out the entire community's health. Also, the system's capability to produce meaningful information about donor behavior, blood utilization, and operational efficiency will allow blood banks to best utilize their resources, pinpoint problem areas, and maintain an uninterrupted flow of blood for the patients.

Also, the fact that it interfaces with other hospital systems and can eventually be expanded to a web based/mobile application makes it even more accessible and useful. With the use of new technologies and innovations, the blood bank management system will grow and change as the needs of blood banks change, and will continue for many years to provide great benefits.

Bibliography

[1] P. Priya, V. Saranya, S. Shabana, and K. Subramani, “The optimization of blood donor information and management system by Technopedia,” *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 3, no. 1, pp. 390–395, 2014.

[2] W. A. V. S. Warnakulasooriya, “Blood Bank & Donor Management System,” Ph.D. dissertation, 2021.

[3] L. Shanmugam, Y. Muniandy, and D. P. Wilson, “Development of a Blood Donor and Seeker Management System for Hospital Use.”

[4] H. D. Das, R. Ahmed, N. Smrity, and L. Islam, “Bdonor: A geo-localised blood donor management system using mobile crowdsourcing,” in *2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT)*, 2020, pp. 313–317.

[5] S. Baş, G. Carello, E. Lanzarone, Z. Ocak, and S. Yalçındağ, “Management of blood donation system: literature review and research perspectives,” in *Health Care Systems Engineering for Scientists and Practitioners: HCSE, Lyon, France, May 2015*. Springer, 2016, pp. 121–132.

[6] C. K. Desik, T. D. Prasad, and M. Kandan, “Blood donation system,” in *AIP Conference Proceedings*, vol. 3075, no. 1. AIP Publishing, 2024.