A Web-Based Blood Management System

M. Arun Prasad

2023

A Web-Based Blood Management System

A dissertation submitted for the Degree of Master of Information Technology

M. Arun Prasad

University of Colombo School of Computing

2023

Declaration

Name of the student: M.Arun Prasad
Registration number: 2018/MIT/003
Name of the Degree Programme: Master of Information Technology
Project/Thesis title: A Web-Based Blood Management System

1. The project/thesis is my original work and has not been submitted previously for a degree at this or any other University/Institute. To the best of my knowledge, it does not contain any material published or written by another person, except as acknowledged in the text.

2. I understand what plagiarism is, the various types of plagiarism, how to avoid it, what my resources are, who can help me if I am unsure about a research or plagiarism issue, as well as what the consequences are at University of Colombo School of Computing (UCSC) for plagiarism.

3. I understand that ignorance is not an excuse for plagiarism and that I am responsible for clarifying, asking questions and utilizing all available resources in order to educate myself and prevent myself from plagiarizing.

4. I am also aware of the dangers of using online plagiarism checkers and sites that offer essays for sale. I understand that if I use these resources, I am solely responsible for the consequences of my actions.

5. I assure that any work I submit with my name on it will reflect my own ideas and effort. I will properly cite all material that is not my own.

6. I understand that there is no acceptable excuse for committing plagiarism and that doing so is a violation of the Student Code of Conduct.

Signature of the Student	**Date (DD/MM/YYYY)**
M. Arun Prasad	28/09/2024

Certified by Supervisor(s)

This is to certify that this project/thesis is based on the work of the above-mentioned student under my/our supervision. The thesis has been prepared according to the format stipulated and is of an acceptable standard.

	Supervisor 1	**Supervisor 2**	**Supervisor 3**
Name	Mr. G P Seneviratne		
Signature			
Date	28/09/2024		

Preface

The efficient management of blood resources is critical in healthcare. It impacts patient care, safety and the optimization of healthcare delivery. The innovative solution to the age-old problem of blood management is finally here. The system aims to revolutionize blood management by providing healthcare institutions with a comprehensive, user-friendly, and highly efficient platform.

The system draws upon the latest advancements in web-based technology and healthcare management to provide a sophisticated yet intuitive solution. It addresses challenges such as inventory tracking and allocation, transfusion protocols, and regulatory compliance to enhance patient outcomes, streamline operations, and improve cost-effectiveness.

This preface invites healthcare administrators, clinicians, and technologists to participate in this transformative voyage. The goal is to harness the power of technology to usher in a new era of excellence in blood management, where every drop counts, ensuring that every drop matters and that every patient receives the care they merit.

This project uses the modified waterfall SDLC methodology, which provides enhanced adaptability through overlapping phases as needed. WampServer, a web development platform, is utilized, harnessing components like PHP (server-side scripting language), Apache (webserver), and MySQL (database) to construct the system.

Welcome to the dawn of a new era in healthcare. Welcome to the Web-Based Blood Management System.

Abstract

The Web-Based Blood Management System (WBMS) is a comprehensive platform designed to streamline and improve the management of blood resources in healthcare institutions. With a focus on efficiency, transparency, and accessibility, WBMS offers a user-friendly interface accessible through standard web browsers, allowing healthcare professionals to manage blood inventory, donations, transfusions, and related processes efficiently.

WBMS offers comprehensive modules to streamline blood donation processes, including donor and blood depot registration, inventory management, blood screening, transfusion requests, and reporting. An innovative Reward Points system incentivizes donors to give blood while also serving as a form of payment. Healthcare organizations can easily track blood products in real-time with the inventory management module, minimizing waste and ensuring optimal utilization. The donor registration module facilitates the screening and registration of donors, ensuring regulatory compliance and maintaining a reliable pool of donors.

The system facilitates efficient communication between healthcare providers, allowing them to submit and track transfusion requests seamlessly.

WBMS's reporting capabilities provide valuable insights into blood utilization patterns, donor demographics, and inventory levels, empowering healthcare administrators to make informed decisions and optimize resource allocation. Furthermore, the system provides robust security measures to protect valuable patient data and ensure full compliance with stringent data protection regulations.

Overall, WBMS represents a significant advancement in blood management technology, offering healthcare institutions a versatile, user-friendly, and efficient solution to optimize blood resource utilization, enhance patient safety, and streamline workflow processes.

Acknowledgement

Completing this project owes much to the contributions of various individuals, and their assistance is sincerely appreciated. The guidance and support provided by the MIT Coordinators, University of Colombo School of Computing (UCSC) staff, and Project Examination Board proved invaluable in facilitating the learning and application of system development life cycle concepts.

Gratitude is extended to the UCSC community for their ongoing guidance, supervision, and provision of project-related information. Special thanks are reserved for UCSC lecturers, particularly Mr G P Seneviratne, for their steadfast support and guidance throughout the endeavour.

Recognition is also due to the colleagues Dr. Pio Wilfred and Adrian Balthazar for their assistance. Lastly, acknowledgement is extended to family and friends for their consistent support and cooperation. The collective efforts of all involved, both directly and indirectly, are deeply appreciated.

Table of Contents

Declaration ..... **iii**
Preface ..... **v**
Abstract ..... **vi**
Acknowledgement ..... **vii**
Table of Contents ..... **viii**
List of Figures ..... **xii**
List of Tables ..... **xiv**
List of Acronyms ..... **xv**
Chapter 1 : Introduction **1**
1.1 Problem Domain: 1
1.2 Motivation for the project: 1
1.3 Aims, and Objectives of the project: 2
1.4 Scope of the project 4
1.5 Structure of Dissertation 6
Chapter 2 : Background **8**
2.1 Introduction 8
2.2 Analyzing the Existing Manual System 8
2.3 Requirement Analysis 10
2.3.1 Key System Features 10
2.3.2 System Authorization 11
2.3.3 Registering Donor 12
2.3.4 Withdraw Registration 14
2.3.5 Register as Blood Depot 15
2.3.6 Blood Donation 16
2.3.7 Blood Request 18
2.3.8 Generate Reports 20
2.3.9 Blood Donor Recipient Mapping 21
2.4 Existing Similar Systems 21
2.4.1 Haemonetics' Blood Center Business 21

2.4.2 Other Similar Systems 23
2.5 Requirements Discovery Methods 25
2.6 Non-Functional Requirements of the proposed system 25
2.7 Process Model 27
2.7.1 Modified Waterfall 27
2.8 Summary 28
Chapter 3 : DESIGN 29
3.1 Introduction 29
3.2 Design Architecture 29
3.3 Web Based System Solution 30
3.4 UML Diagrams 31
3.4.1 Use Case Diagram of the proposed System 32
3.4.2 Sequence Diagram of the System 33
3.5 ER Diagram and the Database Design 34
3.6 Data Base Normalization 34
3.7 Interface Design 36
3.8 Summary 43
Chapter 4 : IMPLEMENTATION 44
4.1 Introduction 44
4.2 System Architecture 44
4.3 Implementation Environment 45
4.3.1 Hardware Environment 45
4.3.2 Software Environment 46
4.4 Developments Tools and Used Technologies 47
4.5 Key codes, validations, and reused snippets 48
4.5.1 Template for Dynamic Web Pages 48
4..2 Authenticate user sessions 49
4.5.3 Usages of Cookies and Session 50
4.5.4 Log out Script with destroying the stored Session and Cookies 50
4.5.5 Prevent from SQL Injections 51
4.5.6 Core Reusable Functions 52
4.5.7 Java Script Validation 55

4.5.8 PHP Validation 56
4.5.9 Data Base Triggers 57
4.5.10 Other Reused Codes 58
4.6 Deployment Summary 59
4.7 Summary 59
Chapter 5 : EVALUATION 60
5.1 Introduction 60
5.2 Testing Objectives 60
5.2.1 Manual Testing: 60
5.2.2 Automated Testing (BDD Cucumber with Selenium WebDriver): 61
5.3 Testing Procedures 61
5.3.1 Manual Testing Procedures: 61
5.3.2 Automated Testing Procedures: 61
5.4 Usability Report 62
5.4.1 Usability Testing Results: Feedback from Usability Testing Sessions: 62
5.4.2 Evaluation of User Interface Design: 62
5.4.3 Overall User Experience (UX) Evaluation: 62
5.4.4 Automated Testing Evaluation Effectiveness of Automated Testing with BDD 62
5.5 Test Cases / Results 64
5.5.1 Login Validations for Donor/Requestor User Logins 64
5.5.2 Register User Validations and Test Cases/Results 65
5.5.3 Automated Selenium Testing for Blood Depot Status Changes with BDD 67
5.6 Summary 70
Chapter 6 : CONCLUSION 71
6.1 Introduction 71
6.2 Work Carried Out 71
6.3 Limitations 72
6.4 Assumptions Made 72
6.5 Lessons Learned 72
6.6 Summary 73
REFERENCES 74
APPENDIX A 75

APPENDIX B **81**
APPENDIX C **88**
APPENDIX D **91**
APPENDIX E **95**
Project Gannt Chart 95

List of Figures

Figure 1 Haemonetics Global Presence 22
Figure 2 Modified WaterFall Model 27
Figure 3.1 Use Case diagram of proposed system 32
Figure 3.2 Sequence diagram of Proposed Solution 33
Figure 3.3 ER diagram of the system 34
Figure 3.4 initial login screen 37
Figure 3.5 User registration interface. 38
Figure 3.6 Admin User Login interface 38
Figure 3.7 Blood Depot Activation Initial Screen 39
Figure 3.8 Blood Depot Activation Detail Screen 39
Figure 3.9 Management Report Interface 40
Figure 3.10 All Users Report 40
Figure 3.11 User picture upload 41
Figure 3.12 Blood Depot User Host Campaign Screen 41
Figure 3.13 Donor/ Recipient User Login screen 42
Figure 3.14 Donor/ Recipient User Home screen 42
Figure 3.15 Health Assessment Questionnaire prior to Blood Donation 43
Figure 4.1 Flow of interaction of the system 45
Figure 4.2 Dynamic web pages creation template 48
Figure 4.3 Code used to authenticate User access to the application (Login page) 49
Figure 4.4 Session start code 50
Figure 4.5 Logout script 50
Figure 4.6 SQL injection prevention techniques 51
Figure 4.7 getRequesterBloodGroup function 52
Figure 4.8 getRequesterDonor function 53
Figure 4.9 User module body.php 54
Figure 4.10 JavaScript validations 55
Figure 4.11 Date, NIC validation - PHP function 56
Figure 4.12 Email, Phone validations - PHP functions 57
Figure 4.13 Database Trigger (User_Master_Tbl) 57
Figure 4.14 Database Trigger (blood_depot_master_tbl) 58
Figure 5.1 Sample BDD Feature File 67
Figure 5.2 Sample Step Definition File 68
Figure 5.3 Sample Test Runner File 68
Figure 5.4 Sample Test Runner File 69
Figure A.1 WAMP Installation first step window 75
Figure A.2 WAMP Installation second step window 75
Figure A.3 WAMP Installation third step window 76
Figure A.4 WAMP Installation fourth step window 76
Figure A.5 Change root password of database 77

Figure A.6 Deploy application sources....77
Figure A.7 Sendmail folder deployment....78
Figure A.8 Sendmail.ini modification....78
Figure A.9 MySQL workbench installation first step window....79
Figure A.10 MySQL workbench installation second step window....79
Figure A.11 Database Import First step window....80
Figure A.12 Database Import second step window....80
Figure B.1 Admin master table structure....81
Figure B.2 Admin Login table....81
Figure B.3 User master table....82
Figure B.4 User Login table....82
Figure B.5 Blood_depot_master table....83
Figure B.6 Blood_depot_login table....83
Figure B.7 Donor table....84
Figure B.8 Donation Campaign table....84
Figure B.9 Blood Inventory table....85
Figure B.10 Blood Request table....85
Figure B.11 Blood Donation Event table....86
Figure B.12 Reward Transaction table....86
Figure B.13 Reward Balance table....87
Figure C.1 User picture upload....88
Figure C.2 User password email generation....89
Figure C.3 Random password on New User Creation....89
Figure C.4 JavaScript validations....90
Figure C.5 User registration java script validations....90
Figure D.1 Home Page - Reports....91
Figure D.2 All User Report....92
Figure D.3 All Blood Depot report....92
Figure D.4 All Active Blood Campaigns report....93
Figure D.5 Pending Not Fulfilled report....93
Figure D.6 Eligible Donor report....93
Figure D.7 Donation Match report....94
Figure D.8 Transaction report....94
Figure D.9 Eligible Users for Reward Points Redemption report....94
Figure E.1 The project's Gantt Chart illustrates the project's timeline and tasks.....95

List of Tables

Table 2.1 Blood Donor Recipient Mapping.....21
Table 4.1 Deployment Summary 59
Table 5.1 Test cases/results for user login validation 65
Table 5.2 Test cases/results for add user module validation.....66

List of Acronyms

CSS - Cascading Style Sheet

GB - Giga Byte

GHz - Gigahertz

HTML - Hypertext Mark- up Language

NIC - National Identity Card

PC – Personal Computer

PHP - Hypertext Pre- processor

RAM - Random Access Memory

SQL - Structured Query Language

TB -Tera Byte

WAMP -Windows, Apache, MySQL, PHP

WWW - World Wide Web

WBMS- Web Based Management System

Chapter 1 : Introduction

The introduction contains details of the project's motivation, goal, aim, and objective. This chapter will give a summary of the problem and a solution to it.

1.1 Problem Domain:

Effective blood management is vital for ensuring optimal patient care, mainly when blood transfusions are necessary. The process begins with doctors assessing whether a patient requires a blood transfusion and, if so, determining the most suitable blood type for that individual. Any delays in procuring the correct blood for a patient can have severe consequences, potentially resulting in loss of life.

Traditional blood donation campaigns often involve cumbersome data entry processes and manual paperwork when registering willing blood donors. Moreover, retrieving relevant donor data can be time-consuming when a patient urgently requires blood. Additionally, reliance on manual repositories may lead to the availability of outdated information, which is unacceptable given the critical nature of the process.

An adequate blood supply can only be assured through voluntary blood donation. A large proportion of adult populations had poor knowledge, unfavorable attitudes, and low practice toward voluntary blood donation. Therefore, local and national blood banks and transfusion agencies should design strategies that can be implemented to improve the knowledge or attitude of the adult population and motivate the population to donate blood voluntarily.(Mussema et al., 2023)

1.2 Motivation for the project:

A web-based blood donation system is necessary to address these challenges effectively. This system aims to streamline the matching of blood donors with recipients by collecting and maintaining comprehensive blood group details from various donors and efficiently distributing them to those in need. By eliminating paperwork and reducing delays inherent in traditional blood donation campaigns, the proposed system provides timely assistance to patients in critical

need of blood. Ultimately, this system has the potential to be a life-saving resource for individuals facing urgent medical situations requiring blood transfusions.

It has been observed that people tend to be less inclined towards donating blood unless they have a personal connection, such as a close relative or friend, in need of blood. To address this issue, a proposed system aims to enhance the motivation of donors by offering them incentives, which can encourage them to donate blood more frequently. The objective is to create a framework that can improve the overall blood donation rate and ensure a steady supply of blood for those in need.

1.3 Aims, and Objectives of the project:

The project aims to revolutionize blood donation management by establishing a leading blood collection repository accessible to most ordinary people in the country. The proposed application introduces several innovative features:

- **Partnership with Stakeholders:** The project collaborates with hospitals and blood collection centres to ensure widespread accessibility and robust storage of blood records, setting a new standard in blood donation infrastructure.
- **Web-Based Donation Platform:** Introducing a user-friendly web-based platform that connects blood donors with recipients seamlessly, offering a novel approach to fastening the gap between supply and demand for blood.
- **Nationwide Campaigns and Drives:** Organizing nationwide blood donation campaigns in collaboration with stakeholders demonstrates a proactive approach to maintaining an adequate blood supply, distinguishing the project's initiative from conventional donation efforts.
- **Comprehensive Management System:** The comprehensive blood donation management system ensures efficient matching and distribution of blood by maintaining accurate donor and recipient details, setting a benchmark for precision and effectiveness in blood management.

- **Recipient Tracking and Reward Integration:** Enabling recipients to track and search for suitable donors easily and facilitating seamless reward processes adds a layer of convenience and transparency, setting a precedent for donor-recipient interaction.
- **Promotion of Voluntary Donation:** The project promotes voluntary blood donation by providing accessible information, fostering awareness, and showcasing a commitment to community engagement and participation.
- **Confidentiality and Security:** Emphasizing confidentiality, convenience, and security in all blood donation activities within the system sets a new standard for privacy and trust in blood donation management platforms.
- **Administrative Support:** Providing administrators with tools for generating necessary reports and conducting day-to-day operations ensures smooth and effective blood management, demonstrating a holistic approach to system functionality.
- **Donor Engagement:** Implementing self-registration and withdrawal features and alert systems to encourage donation showcases a proactive approach to donor engagement and retention.
- **Emergency Response:** Enabling rapid response during emergencies by alerting potential donors and coordinating mobile units prioritizes efficiency and effectiveness in critical situations.
- **Incentivization and Rewards:** Introducing a rewards system to incentivize regular donors highlights a unique approach to enhancing donor retention and encouraging consistent blood donations.
- **Convenience for Donors and Blood Depots:** Providing easy access to information about nearby blood drives or centres enhances convenience for walk-in donors, showcasing a commitment to donor-centricity and accessibility and Eliminate the Manual paperwork

Overall, the project's multifaceted approach to blood donation management presents a novel and innovative solution that addresses various aspects of the donation process, setting a new standard for efficiency, accessibility, and community engagement in blood donation initiatives

1.4 Scope of the project

The scope of the Web-Based Blood Donation system is outlined below, covering a range of functionalities to address challenges in the traditional blood donation process effectively:

Registration of Donors, Recipients, and Blood Depots:
Enable donors and recipients to register within the system.
Allow hospitals and blood depots to enroll as participants.

Blood Donation Campaigns and Drives:

- Organize nationwide blood donation campaigns and drives to foster donations.
- Permit blood depots to host blood drives for defined periods.

Urgent Blood Requests and Monitoring:

- Enable recipients to urgently request specific blood groups from designated hospitals or blood depots through the system.
- Provide an option to monitor all pending urgent blood requests.

Emergency Response and Donor Coordination:

- Provide an option to view all urgent pending blood requests.
- Identify suitable donors during emergencies and coordinate mobile blood drives to collect blood from them.

Comprehensive Donor Management:

- Maintain comprehensive and up-to-date details of blood donors, including relevant certificates.
- Enable donors to obtain Donation IDs and complete their donor questionnaires digitally before attending blood donation campaigns.
- Enable blood depots and administrators to review and approve digitally completed questionnaires/ Blood Donation Requests.

Donation Acceptance and Management:

- Facilitate administrators and blood depots in accepting donations and sending email alerts to donors and recipients correspondingly.

Blood Group and Stock Management:

- Manage and track blood groups and stock details for effective blood management.

Alert Notifications:

- Dispatch email alerts to eligible registered donors for their involvement in scheduled blood donation campaigns.
- Share user creation details/passwords via email.

Withdrawal and Expire Options:

- Offer donors the option to withdraw their registration from the system.
- Enable administrators to expire the donor status of users failing to comply with requirements.

Donation/Request History and Reward Points Tracking:

- Track and exhibit the donation history of each donor, attributing points for each donation made.
- Provide a provision to utilize accrued points to redeem blood in times of need.

User Management/ Personal Details Modification:

- Allow donors, recipients, and blood depots to modify their details, such as address, email address, and contact number.
- Allow administrators to remove bogus users from the system.

Real-Time Availability Checking:

- Provide real-time availability checking for specific blood groups.

Management Information System (MIS) Reports:

- Generate and display relevant MIS reports for donors, recipients, hospitals, blood depots, and administrators, aiding in monitoring and decision-making.

Location-Based Campaign Planning:

- Utilize MIS reports to identify locations/areas with a high availability of willing blood donors, facilitating the organization of targeted blood donation campaigns.

By implementing these functionalities, the Web-Based Blood Donation system aims to streamline and enhance the blood donation process, ensuring efficient coordination between donors, recipients, hospitals, and blood depots while addressing critical needs during emergencies.

1.5 Structure of Dissertation

This project dissertation can be categorized into six major chapters. The first chapter is about the introduction to web-based blood management system.

Chapter 02 – Background

This chapter contains the details of the existing manual system. How are blood donations carried out now by the existing manual system, and what are the functional and non-functional requirements? How fact gathering techniques and UML diagrams were used to analyze the requirements

Chapter 03 - Design

This chapter contains the details of the proposed solutions to achieve the requirements in the analysis chapter. It contains details of database structure and user interface design. Database diagram, UML diagrams are included here.

Chapter 04 – Implementation

This chapter contains the details of development tools which are used and the hardware software requirements and implementation environment of the system. And it also contains the important codes and reused codes details.

Chapter 05 –Testing and Evaluation

This chapter contains the details of how the developed system has been tested. And it also contains the test cases and test results and client evaluation details.

Chapter 06 – Conclusion

This chapter will conclude the dissertation with evaluation of the system whether project objectives are met or not. It contains the details about the lessons learnt from project and future enhancement to the system.

Chapter 2 : Background

2.1 Introduction

This chapter will provide a comprehensive overview of the literature review, background information on the problem, and the problem statement. It will delve into the extensive landscape of blood management software, which is utilized by various organizations across different geographical locations.

2.2 Analyzing the Existing Manual System

The current Blood Donation system in Sri Lanka, which relies heavily on manual processes, often leads to significant delays due to paperwork and a lack of incentives for donors. This results in individuals being primarily motivated to donate if they have a personal connection to someone in need.

The reliance on manual paperwork and the absence of digital connectivity in the blood donation process poses significant challenges, especially during emergencies such as accidents, cancer cases, or pediatric illnesses. The perishable nature of blood further complicates the situation, necessitating a systematic approach to managing critical information. Developing a shared Blood Repository across all depots and banks is necessary to address these challenges and ensure a more efficient and effective blood donation system.

Our proposed system aims to streamline the process by centralizing blood bank data and donor information. A centralized system will efficiently assist hospitals and recipients who need specific blood types. The benefits of this system are manifold, including prompt donor matching with those in need, thereby mitigating the issue of delays and significantly increasing the number of potential donors.

User Classes and Characteristics

In the Blood Management System, three primary user roles are interacting with the system:

Donors/Recipients: Individuals who donate blood or receive blood donations.
Blood Depots/Hospitals: Entities responsible for managing blood storage, distribution, and transfusion, including hospitals and blood depots.
Blood Management System Admin Users: Administrators overseeing and managing the Blood Management System are responsible for activities such as user management, system configuration, and smooth operation.

Who can donate blood?

The person must fulfill several criteria to be accepted as a blood donor. These criteria are set forth to ensure the safety of the donor as well as the quality of donated blood.(“Donate Blood,” n.d.)

Type of Donors

- Voluntary non remunerated donors. (Donate for the sake of others and do not expect any benefit. their blood is considered safe and healthy)
- Replacement donors. (donate to replace the units used for their friends or family members)
- Paid donors. (Receive payment for donation)
- Directed donors. (donate only for a specific patient's requirement) (“Donate Blood,” n.d.)

Donor Selection Criteria

- Age above 18 years and below 60 years.
- If previously donated, at least 4 months should be elapsed since the date of previous donation.
- Hemoglobin level should be more than 12g/dL. (this blood test is done prior to each blood donation)
- Free from any serious disease condition or pregnancy.

- Should have a valid identity card or any other document to prove the identity.
- Free from "Risk Behaviors" (“Donate Blood,” n.d.)

2.3 Requirement Analysis

2.3.1 Key System Features

- **User Registration**: The process for individuals to sign up as users of the blood donation system, whether as users(donors/recipients), administrators, or blood depot managers.
- **Register Donor**: Allows individuals to sign up as blood donors in the system.
- **Register Blood Depot:** Registration for blood depots where blood can be stored and managed.
- **Admin Accepts Blood Depot Request**: Administrative action to approve requests from blood depots.
- **Login to Donor/Admin/Blood Depot Profiles**: Allows registered users to log in with their respective profiles.
- **Edit Donor/Blood Depot Details**: Functionality to update personal or organizational information.
- **View Active Campaigns/Pending Requests**: Access ongoing blood donation campaigns and requests awaiting approval.
- **Donate Blood**: Option for donors to give blood.
- **Fill out the Pre-Donation Questionnaire:** Complete the questionnaire before donating blood.
- **Request Blood**: Allows users to request blood when needed.
- **Donation/Request History**: Record of past blood donations and requests made.
- **Reward Points**: System for awarding points to users for their contributions.
- **Host Blood Drive:** Organizing events for blood donation.
- **Organize mobile blood donation drives:** Arranging blood donation events at different locations.
- **Edit Campaigns:** Ability to modify details of blood donation campaigns.

- **Accept Donation:** Confirmation of received blood donations.
- **Manage Stock:** Inventory management of available blood units.
- **Withdraw Donor (Self/Admin):** This option allows a donor to be removed from the system, either by the donor or by an administrator.
- **View Pending Requests:** Access to pending blood donation requests.
- **Cancel the Donation Request before Scheduled Blood Donation:** Allows cancellation of scheduled blood donations.

2.3.2 System Authorization

Description and Priority

End Users must be authenticated before they access the system and start interacting with it. This module would assist system administrators, Blood Depot users donors, and recipients in Logging in to the system and accessing other modules allocated according to their user type. The system facilitates User changes of password and updating of necessary personal details whenever required. Priority of this module is categorized as “High”.

Functional Requirements

Use Case 1:	Login
Actors:	System Administrator/ Blood Depot/ Donor/ Recipient
Description:	Login to Blood Donation System
Pre-Conditions:	None
Main Scenario:	1. Launch the Blood Donation Site from the browser. 2. Select the Respective Login Option (Eg:- Login as ‘Administrator’/ User/ Blood Depot’ 3. User Logs into the Application 3.1. Key in the User Name and password 3.2. Click on the Submit button 4. System Authenticate the user 5. System redirects to Main Home Page after Login

Alternate Scenario	4.(a). Authentication Fails 4(a)1. Prompt Message displaying that Username/ Password is incorrect 4(a)2. Click Ok 4(a)3. Reenter the Password

Use Case 2:	Change Details
Actors:	System Administrator/ Blood Depot/ Donor/ Recipient
Description:	Change/ Update existing user details
Pre-Conditions:	User Successfully Logged into the Application
Main Scenario:	1. Navigate to Change Details Option 2. User Changes required modifiable fields/ file documents in the system 3. Click on the “Submit” button 4. System Validates and Prompts “Do you want to Apply the Changes.” 5. On successful Update System Prompt “Update Successful 5. System reloads back to with the Updated Details
Alternate Scenario	4(a) Validations failed during the changes 4(a)1. Prompt Message displaying that “Validations failed” 4(a)2. Click Ok 4(a)3. Reload to the “Change Details” page with the sticky data already keyed in

2.3.3 Registering Donor

Description and Priority

As a prerequisite, Donors should register with the system and upload a certified certificate from the certified doctor before donating Blood. When a new donor is willing to donate Blood, he/she should first register with the Blood Donation system by entering his/her personal/contact details and blood group details into the system. This module facilitates the capture of all required details from the donor before registering him as a donor. Priority of this module is categorized as “High”.

Functional Requirements

Use Case 3 :	Register as User
Actors:	Donor/ Recipient
Description:	Register new donor/recipient into Blood Donation System with his personal /contact/ blood group details
Pre-Conditions:	None
Main Scenario:	1. Launch the Blood Donation Site from the browser. 2. Click on “Register as User” Option 3. Fill all the mandatory details 4. Submit the request 5. The System Validates the request and assigns the user with a unique User ID 6. System send the Login Credentials via Email 7. System redirects to the Main Home Page Interface
Alternate Scenario	5.(a). Validation failed during the registration. 5(a)1. Prompt Message displaying that “Validations failed.” 5(a)2. Click Ok 5(a)3. Reload to the “Registration” page with the sticky data already keyed in.

Use Case 4:	Register as Donor
Actors:	Donor
Description:	Register new donor into the Blood Donation System with blood certificate details and the last donation date details.
Pre-Conditions:	The user Successfully Logged into the Application.
Main Scenario:	1. Navigate to the “Donor Registration” Option. 2. Click on “Registration” Option 3. Fill all the mandatory details 4. Upload the Doctor's Certificate to the system 5. Submit the request

	6. The System Validates the request and process the record 7. System redirects to the Main Home Page Interface
Alternate Scenario	6.(a). Validation failed during the registration 6(a)1. Prompt Message displaying that “Validations failed” 6(a)2. Click Ok 6(a)3. Reload the “Donor Registration” page with the sticky data already keyed in.

2.3.4 Withdraw Registration

Description and Priority

Registered Users can opt out of the Donors list in the Blood Donation System. Thus, the withdrawal registration module would cater to such users' requirements. If Donors are not eligible to donate blood their medical certificate is expired. It needs to be renewed, or if a donor has not provided a valid medical certificate, System Administrators can withdraw some registered donors and mark them inactive or from the donor's list. Priority of this module is categorized as "Medium".

Functional Requirements

Use Case 5:	Withdraw Registration
Actors:	System Administrator/ Donor
Description:	Donor withdraws his registration from Blood Donation System or System Administrator Withdraw registrations of relevant donors who failed to submit the medical report annually.
Pre-Conditions:	The Admin user Successfully Logged into the Application.
Main Scenario:	1. Navigate to “Donor Registration” Option 2. Click on “Withdraw Registration” Button 3. Click ok to confirm the withdrawal. 4. System redirects to Main Dash Board Interface

Alternate Scenario	1. Navigate to the “Change User Status” Option 2. Key In the User ID or ID Number to Search for the user And click on Block 3. Click on Withdraw Registration Button 4. System Prompts, “Do you want to Block the User?” 3. Click ok to confirm the withdrawal. 4. System redirects to Main Dash Board Interface

2.3.5 Register as Blood Depot

Description and Priority

Blood banks and hospitals must be able to register as blood depots within the System. Once validated by the system administrator, the blood depot will be activated, and login credentials will be issued to the Institutions. After Registering, these Blood depots could Host Blood Donation Campaigns, and the System would maintain the Blood Stock for the Registered Blood Depots. Priority of this module is categorized as "High".

Functional Requirements

Use Case 6:	Register Blood Depot
Actors:	Blood Depots
Description:	Register new blood depot into Blood Donation System
Pre-Conditions:	None
Main Scenario:	1. Launch the Blood Donation Site from the browser. 2. Click on “Register as Blood Depot” Option 3. Fill all the mandatory details 4. Submit the request 5. The System Validates the request and assigns the user with a unique Blood Depot User ID 6. Blood Depot would wait until the Administrator activates the Account

Alternate Scenario	5.(a). Validation failed during the registration. 5(a)1. Prompt Message displaying that “Validations failed.” 5(a)2. Click Ok 5(a)3. Reload to the “Registration” page with the sticky data already keyed in.
Post-Conditions:	System Administrator Activates the Blood Depot Registration Request

Use Case 7:	Activate Blood Depot Registration
Actors:	Administrator
Description:	Accept the New Blood Depot Registration
Pre-Conditions:	The admin user Successfully Logged into the Application.
Main Scenario:	1. Navigate to the "Depot Status Update" Option 2. Key in the Blood Depot ID 3. Click on the Manage BD Button 4. Activate the Blood Depot Registration 5. Upon Activation, the System emails the Login User Credentials to Blood Depot's Registered Email.

2.3.6 Blood Donation

Description and Priority

The Blood Donation module is a critical component of the Blood Donation System. Donors can donate blood in response to a specific request ID or participate in campaigns organized by Blood Depots. A Donation ID is generated and provided to the donor upon expressing interest in donation. Before donation, the donor must complete a pre-donation questionnaire in the System. Subsequently, the donor can visit an active Blood Campaign, provide their donor ID, and proceed with the donation.

In emergencies, the System Administrator will host a Mobile Donation Campaign and dispatch the Mobile Blood Unit to nearby locations to facilitate faster donations.

A Donation Completion Email notification will be sent to the donor after the donation.
This module is prioritized as "High" importance.

Functional Requirements

Use Case 8:	Donate Blood
Actors:	Donor
Description:	Donor donates blood to Blood Donation System.
Pre-Conditions:	The logged-in user must meet the eligibility criteria to donate blood.
Main Scenario:	1. Access the "Active Campaign" option from the menu. 2. Click on the "Donate" button to express interest. 3. Retrieve the Donation ID provided. 4. Navigate to the "Donate Blood" section and enter the Donation ID. 5. Complete the Pre-Donation Questionnaire. 6. Submit the information.
Alternative Scenario	1(a). Access the "Pending Request" option from the menu. 2(a). Click on the "Donate" button associated with the Request ID to indicate interest.
Post-Condition:	The donor must visit the campaign location before the campaign period ends.

Use Case 9:	Collect Blood
Actors:	Blood Depot
Description:	The Blood Depot Administrator collects and updates relevant information in the Blood Donation System.
Pre-Conditions:	The donor has completed the blood donation process, and the Donation ID has been shared with the Administrator. Additionally, the System Administrator is logged into the system.
Main Scenario:	1. Navigate to “Accept Donation” Option 2. Key in the “Donation ID”

	3. Validate the details and Collect the Blood from the Donor 4. Confirm the Donation in the system 5. The System Update the Stock details and redirects to the Main Home Page Interface
Post-Condition:	Necessary alert emails are sent from the system to the donor and recipients. Rewards Points are granted to the donor on successful blood donation completion

2.3.7 Blood Request

Description and Priority

The Blood Request feature, a crucial component of the Blood Donation System, empowers individuals needing blood by providing a simple and efficient process. Upon submission, the requester is immediately assigned a unique' Request ID' and can track the status of their request and whether the required blood group is available in the system's stockpile. If the requested blood group is in stock, the requester can swiftly use their redeemable reward points to receive the blood.

When the requested blood group is unavailable in the stockpile, eligible registered donors matching the required blood group (as per the donor/recipient mapping in Appendix D) are promptly notified via email, along with the requester's "Request ID." Donors can then use this "Request ID" to fulfil the Donation Request within the Blood Donation system.

In scenarios where multiple donors enter the same "Request ID," the donation completed first is assigned to the recipient, while other donations are added to the standard Blood Stock. Once the request is fulfilled, the recipient is notified via email, and their redeemable reward points are deducted accordingly. This module is deemed of high priority.

Functional Requirements

Use Case 10:	Request Blood utilizing reward points
Actors:	Recipient
Description:	Recipients in need request a specific blood group using their reward points.
Pre-Conditions:	The recipient successfully logged into the application should have sufficient reward points to request Blood.
Main Scenario:	1. Navigate to the "Request Blood" option. 2. Initiate and Submit a New Request 3. The system displays a "Request ID" with blood availability status 4. Use Redeemable Reward Points to Receive the Blood Donation
Alternate Scenario	3(a). The requested blood is not available in the Stockpile. 3(a)1. The system sends email alerts to eligible registered donors who can donate to the requested blood group to initiate a new blood donation.
Post-Condition:	The system administrator contacts the recipient to arrange blood transition and updates the system records once the blood is issued.

Use Case 11:	Issue Blood
Actors:	Blood Depot Administrator
Description:	If the Blood requested is in stock, the relevant Blood depot issues Blood to the Recipient.
Pre-Conditions:	An Active Pending Request ID should be available to be fulfilled.
Main Scenario:	1. Navigate to “Issue Blood” Option 2. Key in the “Request ID” 3. Confirm the Issuance in the system 4. System redirects to Main Home Page Interface
Post-Condition:	Send Necessary Alert emails from the system to Recipients to Collect the Blood.

2.3.8 Generate Reports

Description and Priority

Donors, recipients, and system administrators will utilize the system to generate or view reports to aid in their daily operations or gain insights into their activities within the system. The system will support administrators in retrieving reports such as:

- All User Report
- All Blood Depots Reports
- Active Campaigns Report
- Donation Match Completed Report
- Pending Not Fulfilled Request
- Transaction Report
- Eligible Donor List
- Available Blood Stock
- Reward Points Redemptions
- Pending Donor Medical Certificate Renewals

The reports module, which is of high priority, will assist in managing various aspects of the blood donation system.

Functional Requirements

Use Case 12:	Generate View Reports
Actors:	System Administrator/ Donor/ Recipient
Description:	Donors, recipients, and administrators generate or view reports according to their user roles' privileges.
Pre-Conditions:	User successfully logged into the application.
Main Scenario:	1. Navigate to “Reports” Section. 2. Click on the relevant report that the user is interested in retrieving. 3. The system displays the report in the body of the webpage.

2.3.9 Blood Donor Recipient Mapping

System would be using the below mentioned mapping to validate the availability of the blood group which could be transfused to the recipient.

Recipient	Donor							
	O-	O+	A-	A+	B-	B+	AB-	AB+
O-	Y	N	N	N	N	N	N	N
O+	Y	Y	N	N	N	N	N	N
A-	Y	N	Y	N	N	N	N	N
A+	Y	Y	Y	Y	N	N	N	N
B-	Y	N	N	N	Y	N	N	N
B+	Y	Y	N	N	Y	Y	N	N
AB-	Y	N	Y	N	Y	N	Y	N
AB+	Y	Y	Y	Y	Y	Y	Y	Y

Table 2.1 Blood Donor Recipient Mapping

2.4 Existing Similar Systems

2.4.1 Haemonetics' Blood Center Business

Haemonetics (NYSE: HAE) is a global healthcare company dedicated to providing a suite of innovative hematology products and solutions for customers, to help them improve patient care and reduce the cost of healthcare. Our technology addresses important medical markets: blood and plasma component collection, the surgical suite and hospital transfusion services.(“Haemonetics Announces Sale Of U.S. Blood Donor Management Software Solution Assets To The GPI Group | Haemonetics Corporation,” n.d.)

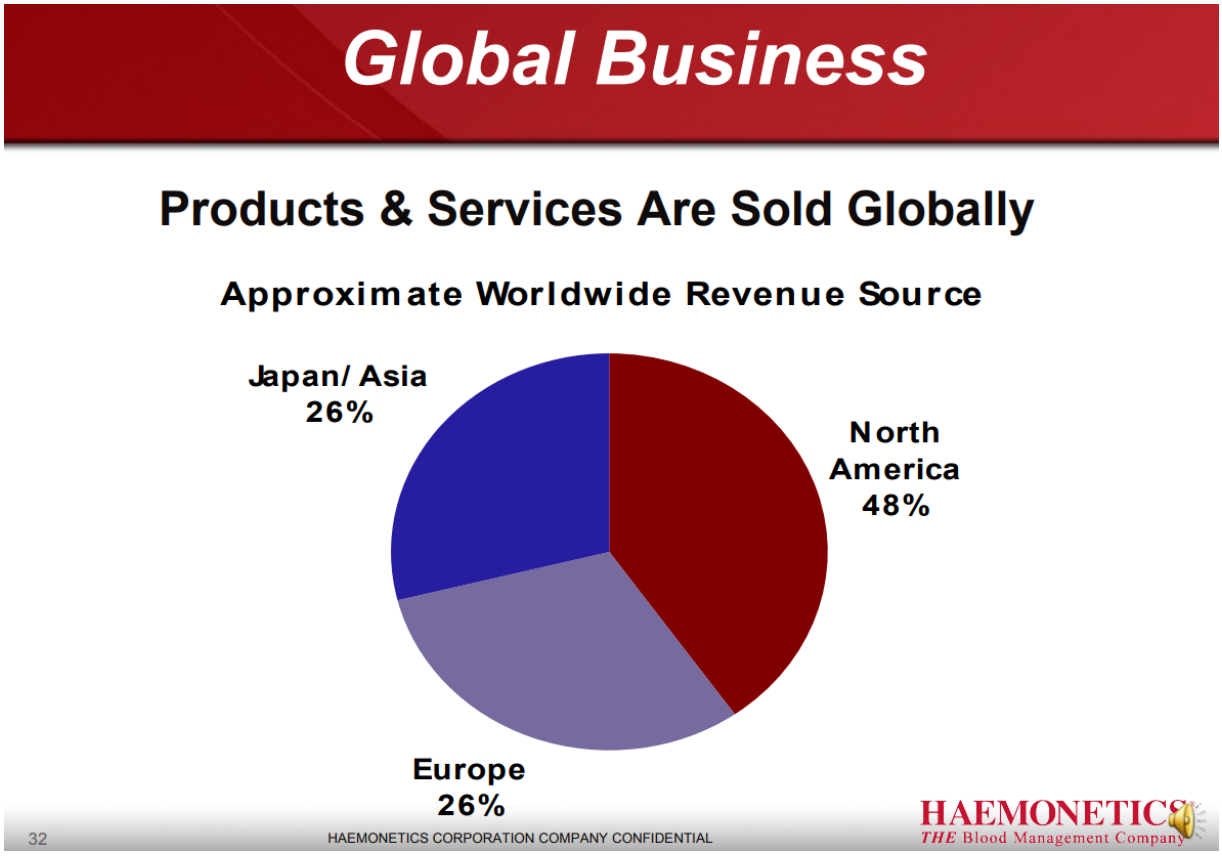
Haemonetics' Blood Center Business is committed to helping blood centers manage blood collections safely, collect the blood components in the greatest demand and attract and retain the right donors. Our comprehensive portfolio enables our customers to provide safe, high-quality blood components to hospitals and their patients around the world.(“Haemonetics® Blood Center Solutions | Manage Blood Collection Safely,” n.d.)

Blood Donor Management Software:

Haemonetics offers blood donor management software designed to streamline and enhance the process of blood donation. This software solution facilitates donor registration, screening, and scheduling, allowing blood centers to efficiently manage donor information and blood ‘'collections. It helps ensure donor eligibility, compliance with regulatory requirements, and timely communication with donors regarding appointments and donation events.

Global Presence:

Haemonetics operates globally, serving customers in over 130 countries worldwide. With a strong presence in North America, Europe, Asia-Pacific, and Latin America, the company collaborates with healthcare providers and blood centers across diverse regions to address the evolving needs of the healthcare industry and improve patient care.



(“Haemonetics Presentation | PPT,” n.d.)

Figure 1 Haemonetics Global Presence

2.4.2 Other Similar Systems

Numerous blood donation systems have been developed to address similar challenges and enhance the blood donation process. Among these, notable examples include:

Name	Place	Year	Project Description
Online Blood Donation Reservation and Management System	Jeddah	2013	Developing a blood MIS to manage the records of the people who need blood so the donors easily can find them and donate to them.
Bharat Blood Bank	India	2005	It allows recipients to reach donors. It created a database of donors, classified by locality. Donors in India who want to donate blood can register, after reading the basic constraints of donating blood. Also, anyone can refer friend(s) by just providing their email IDs. Blood recipients can browse the site and display the list of blood donors who are close to their locality
Online Blood Donation Reservation and Management System	Malaysia	2006	It enables the donors to make online reservation and to know about all the blood donation events.
A Web-based blood donor MIS	Uganda	2009	It enables result and performance monitoring of each blood donation activity in a confidential, convenient, and secure way
Blood Bank Management System	India	2006	Developed a web portal to facilitate the interaction between the demand for blood and provider. This system makes available safe blood and other blood components, which can offer moral assistance, consistent with the long-term welfare-being of the community
Wiqaih	KSA	2012	Linking those wishing to donate with regulators. and which blood banks across the automated system. Is easy to use and both sides can cooperate to achieve the overall vision for system protection.

(Hashim et al., 2014)

Pros/Cons in using the existing software to achieve the Objectives.

Pros:

- Utilizing existing blood donation systems allows for the reuse of established software with the potential for customization to meet specific requirements.
- It saves time and resources by avoiding developing a new system from scratch.
- Some existing systems may already have robust features for blood donation management, such as donor tracking, inventory management, and communication tools.

Cons:

- Existing systems may not have been specifically developed for the Sri Lankan context, requiring localization and customization efforts to align with local needs and practices.
- These systems may lack certain important features for the proposed project objectives.
- Existing systems might have an abundance of unnecessary or unused features that are not relevant to the Sri Lankan context, leading to unnecessary complexities and confusion for end users.
- Customizations and configurations might be extensive and time-consuming, as they would need to align the system with the specific requirements and workflows of the local blood donation process.
- Skilled personnel are in short supply.
- The implementation of the project is associated with substantial costs.

These existing systems have demonstrated the potential of web-based platforms in improving blood donation processes, enhancing communication between donors and recipients, and ensuring efficient blood inventory management.

However, there is room for further innovation and improvement regarding usability, integration with healthcare systems, and data security. The proposed Web-Based Blood Donation System aims to address these gaps by providing a user-friendly, reliable, and secure platform that streamlines the blood donation process, facilitates timely access to blood for recipients, and enhances overall blood management efficiency.

Considering that the existing systems contain numerous functions and configurations that are irrelevant to the Sri Lankan context, it is recommended that a custom software solution be developed. This custom software can be easily configured to meet the specific needs of Sri Lankan users, ensuring a streamlined system that provides significant value to the end users and effectively achieves the project's objectives.

2.5 Requirements Discovery Methods

The following Requirement Discovery techniques were utilized to gather the necessary information for this project:

- Sampling existing documents, reports, forms, databases, and other sources of information was conducted.
- Research of relevant literature was carried out.
- Observation of the current manual Blood Donation Procedures was performed.
- Questionnaires and surveys were distributed.
- Interviews were conducted with the Medical Professionals.

2.6 Non-Functional Requirements of the proposed system

Non-functional requirements are vital to the system's usability and performance, although they are not directly related to its functionalities. Neglecting these requirements could make the system entirely unusable. The system is expected to meet the following non-functional requirements:

Performance Requirements

- 95% of the system responses should be within 5 sec, except for the report downloading, where the download time per report should be at most 45 seconds.
- The system should run on Core I5 machines with free memory space of 20 GB

Security Requirements

- The system should have a proper authentication mechanism to authenticate the Donor, System Administrators and the recipients.
- The system should block the user account temporarily in case three continuous wrong password attempts are registered. In such an event, the user should be forced to change the password via the registered email account.

Software Quality Requirements

Reliability

- The system should be 100% error-free and reliable

Robustness

- The process should continue despite unexpected problems

Usability

- The system should be easy to work with, and anyone with basic computer knowledge should be able to work with the system

Availability

- The system should be available 99% of the time

Maintainability

- The system should be flexible to adapt to future changes and easy to maintain

2.7 Process Model

2.7.1 Modified Waterfall

SDLC methodology "Modified Waterfall" was used for the development, which involves phases such as "Requirements Definition", "Design"," Implementation", "Verification", and "Maintenance". This methodology is more flexible than the pure waterfall model, and it enables the phases to overlap when needed.

Throughout the development process, the phases of the Modified Waterfall methodology can overlap as necessary to accommodate changes or to expedite certain aspects of the project. This approach allows for greater flexibility while still providing a structured framework for successfully developing and delivering the web-based system.

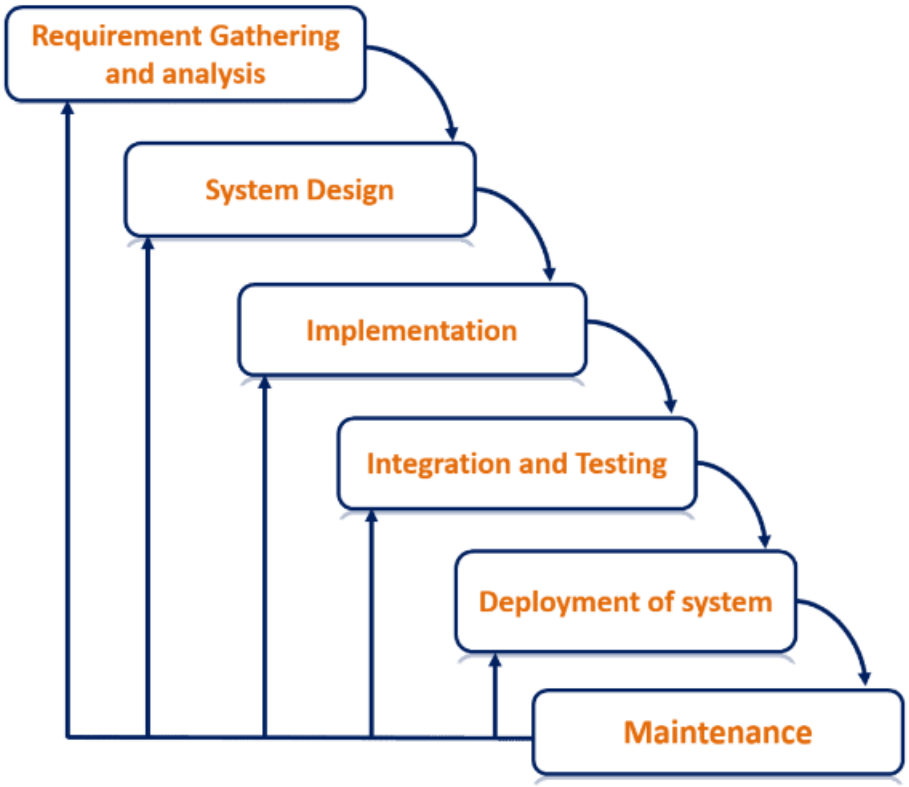


Figure 2 Modified WaterFall Model

2.8 Summary

This Chapter covered the study's background, encompassing an analysis of the existing manual blood management system and the requirements for implementing an automated solution. It explored similar systems, discussed methods for requirements discovery, outlined functional/non-functional requirements, and explained the chosen process model. The subsequent Chapter focused on detailing the design and methodology adopted for the project.

Chapter 3 : DESIGN

3.1 Introduction

In the Design Methodology phase, the system's design is developed based on the requirements and information collected during the requirement-gathering phase. This process involves delineating the system's architecture, specifying its interfaces, and identifying the data that needs to be processed to fulfil the specified requirements. The main goal of this phase is to create a blueprint for the system that can be implemented effectively and efficiently.

This design chapter is a testament to the problem-solving abilities, containing detailed solutions for the challenges encountered in the analysis chapter. It provides a comprehensive overview of the database structure and design, user interface design, and UML diagrams.

3.2 Design Architecture

The web-based blood management system employs a three-layer architecture to effectively manage UI components, business logic, and system data. The presentation layer contains HTML/CSS files for UI components.

Employing a 3-tier architecture enhances system performance and offers greater flexibility in deployment. Because there is no direct connection between the front end and the database, this architecture significantly enhances security, reliability, and ease of maintenance. Modifications in one layer can be implemented without causing major disruptions in other layers.

3-Tier Architecture

A 3-tier application architecture is a modular client-server architecture that consists of a presentation tier, an application tier and a data tier. The data tier stores information, the application tier handles logic and the presentation tier is a graphical user interface (GUI) that communicates with the other two tiers. The three tiers are logical, not physical, and may or may

not run on the same physical server.(“What is a 3-Tier Application Architecture?,” n.d.)

Presentation tier: This tier, which is built with HTML5, cascading style sheets (CSS) and JavaScript, is deployed to a computing device through a web browser or a web-based application. The presentation tier communicates with the other tiers through application program interface (API) calls.(“What is a 3-Tier Application Architecture?,” n.d.)

Application tier: The application tier, which may also be referred to as the logic tier, is written in a programming language such as Java and contains the business logic that supports the application's core functions. The underlying application tier can either be hosted on distributed servers in the cloud or on a dedicated in-house server, depending on how much processing power the application requires.(“What is a 3-Tier Application Architecture?,” n.d.)

Data tier: The data tier consists of a database and a program for managing read and write access to a database. This tier may also be referred to as the storage tier and can be hosted on-premises or in the cloud. Popular database systems for managing read/write access include MySQL, PostgreSQL, Microsoft SQL Server and MongoDB.(“What is a 3-Tier Application Architecture?,” n.d.)

3.3 Web Based System Solution

Developing a Blood Management System as a web-based solution presents a unique set of advantages that set it apart from other solutions:

Convenience and Flexibility: A web-based system offers a user-friendly experience, allowing a broader audience to access it from any device with an internet connection. This adaptability enables the system to grow with the user base, eliminating the need for significant infrastructure changes.

Platform Independence: Web-based systems are not tied to any specific operating system or device, making them highly versatile. Users can access the system regardless of the platform they are using, whether it is Windows, macOS, Linux, or mobile operating systems.

Ease of Maintenance and Updates: Web-based applications can be updated and maintained centrally, reducing the effort required for distribution and installation compared to traditional desktop applications. Changes can be implemented seamlessly without disrupting user experience.

Financial Efficiency: Implementing a web-based solution incurs significantly lower implementation costs than other methods. The system operates within a web browser, so there's no need for additional hardware or software on client devices, making it a cost-effective choice.

Accessibility over Network: The system's web-based nature allows users to access it over a network, facilitating easy information sharing among stakeholders. Whether healthcare professionals access patient data from different locations or administrators manage inventory remotely, the system's accessibility is greatly enhanced.

In summary, opting for a web-based Blood Management System ensures broader accessibility, platform independence, easier maintenance, cost-effectiveness, and enhanced accessibility over a network, making it the ideal solution for meeting the project's functional and non-functional requirements.

3.4 UML Diagrams

UML diagrams like User Case diagram and Class Diagram are used to model the design solution for the proposed System.

ER Diagram is being used to show the relationship between the entities in the system.
With UML diagrams, it is easier to understand the functionalities of the system rather than

having textual description about the system. UML has the ability to implement graphical models that describes the system functionalities.

3.4.1 Use Case Diagram of the proposed System

Below mentioned figure 3.2 illustrates use case diagram of the system.

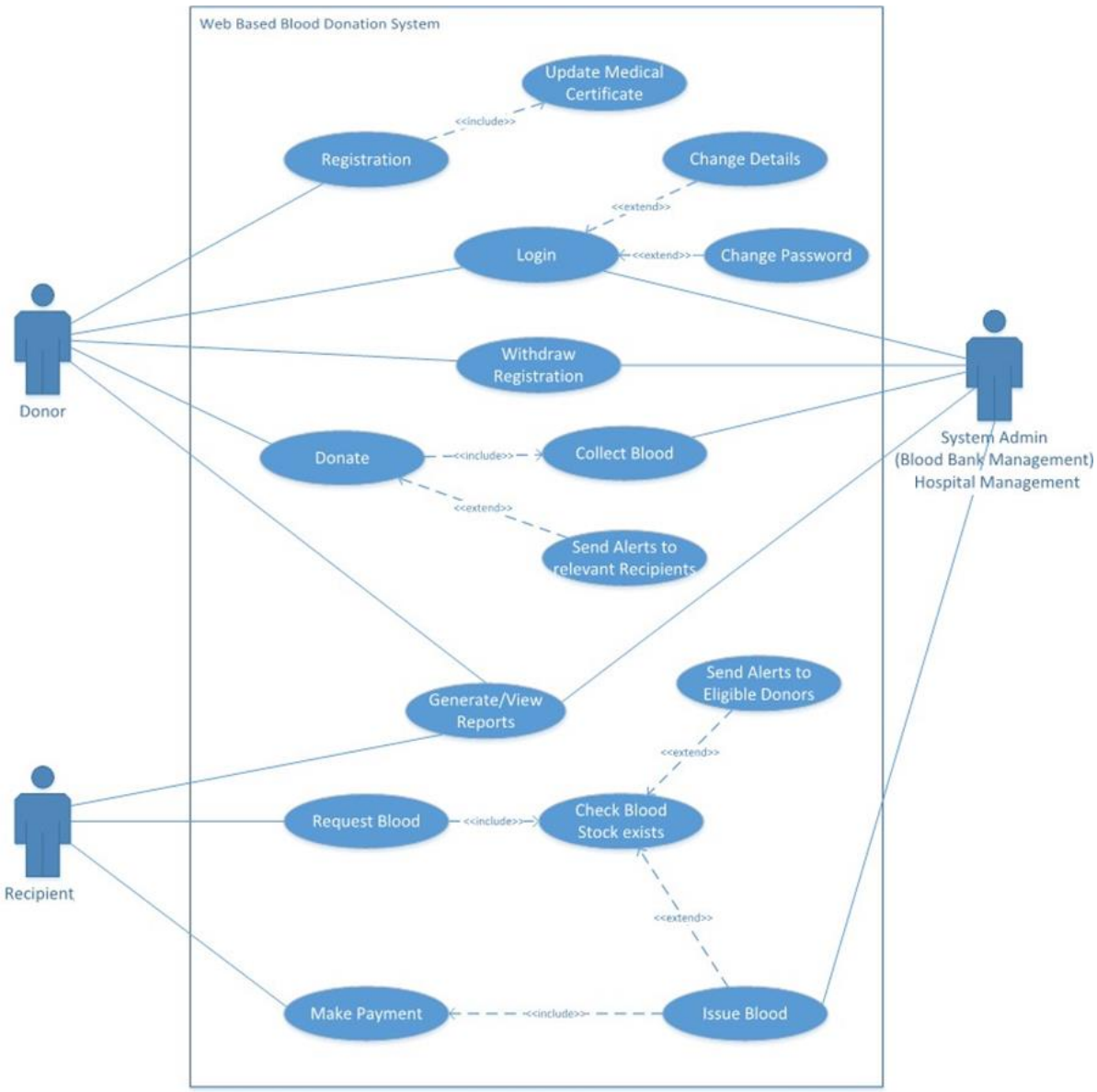


Figure 3.1 Use Case diagram of proposed system

3.4.2 Sequence Diagram of the System

See figures 3.3which depicts sequence diagrams of lending, returning processes

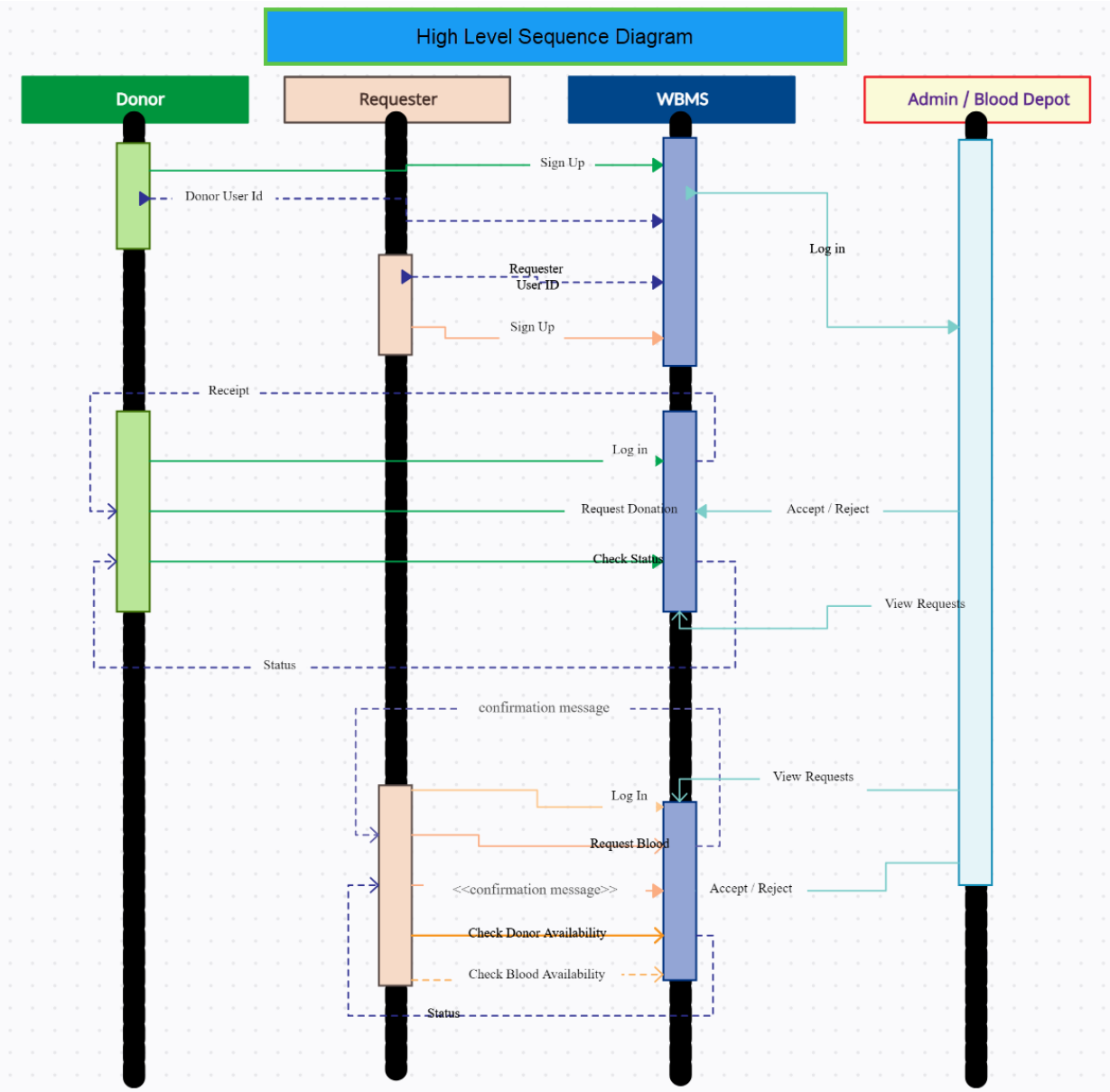


Figure 3.2 Sequence diagram of Proposed Solution

3.5 ER Diagram and the Database Design

ER Diagram (see figure 3.5) has been mainly used to create the database structure and Database Design

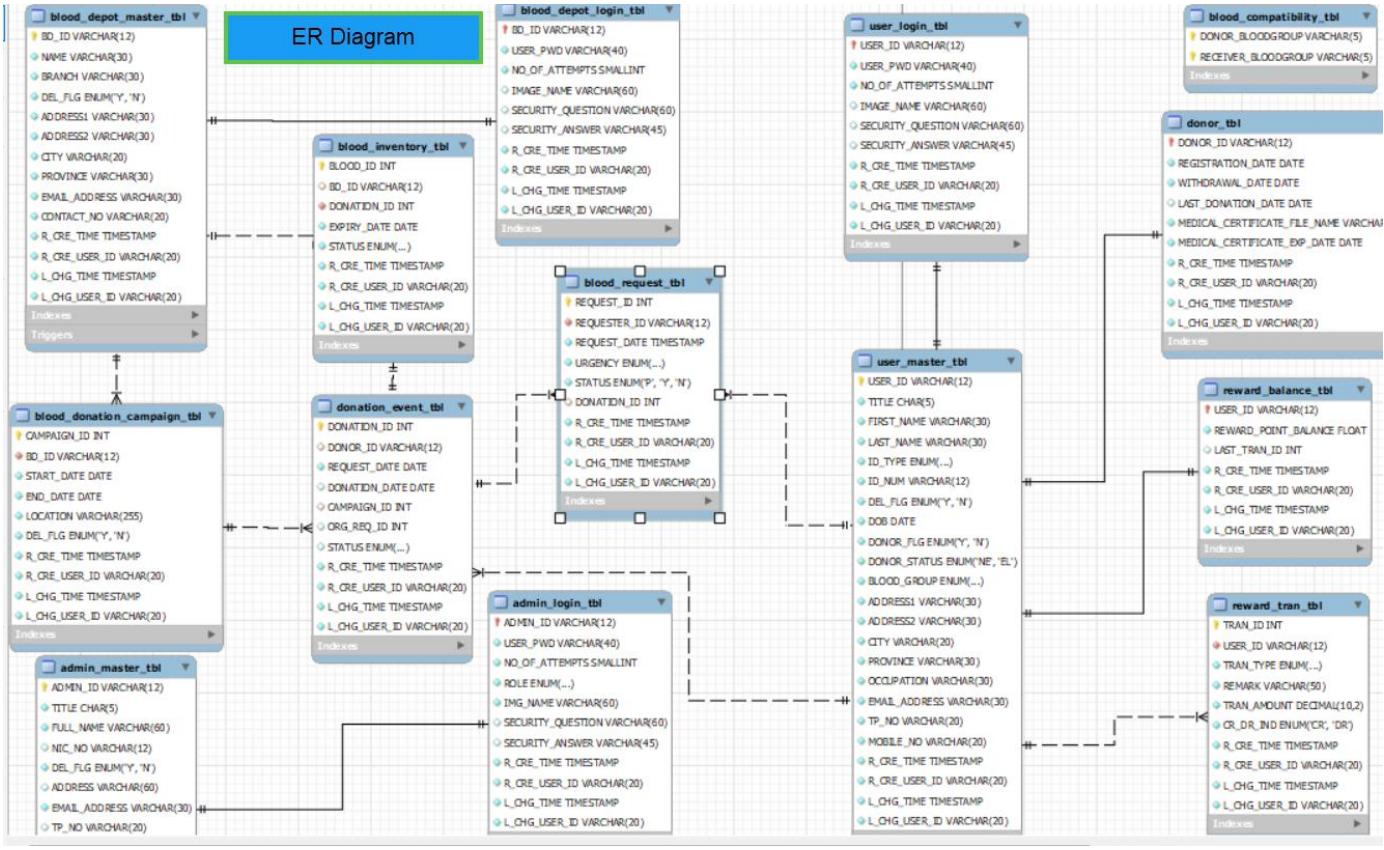


Figure 3.3 ER diagram of the system

3.6 Data Base Normalization

The database design for the Web-Based Blood Management System has been meticulously crafted, focusing on database normalization principles to optimize performance and ensure data integrity. Normalization involves organizing data into tables and establishing relationships to eliminate redundancy and maintain consistency.

First Normal Form (1NF):

Example: Tables such as user_master_tbl and user_login_tbl exemplify the principles of the first standard form by eliminating repeating groups and organizing related data into individual tables. Each table includes a primary key (USER_ID) to identify records uniquely. For instance, the user_login_tbl stores login credentials and associated information for users, while the user_master_tbl contains additional personal details relevant to users. This separation of concerns enhances data organization and facilitates efficient data retrieval.

Second Normal Form (2NF):

Example: The blood_inventory_tbl table separates data related to blood inventory from the donation_event_tbl, thereby avoiding redundant storage of inventory information for each donation event.

It establishes a relationship between the blood_inventory_tbl and donation_event_tbl using a foreign key (DONATION_ID).

Third Normal Form (3NF):

Example: The donor_tbl table eliminates fields not dependent on the primary key (DONOR_ID) to adhere to the third standard form.

Fields such as REGISTRATION_DATE, WITHDRAWAL_DATE, and MEDICAL_CERTIFICATE_FILE_NAME are specific to each donor and are kept in this table, separate from donation events or other unrelated data.

Other Normalization Forms:

The database design primarily focuses on 1NF, 2NF, and 3NF, as adhering to higher normalization forms like BCNF or the fifth standard form may only sometimes be practical. However, the design ensures efficient database utilization, minimizes redundancy, and enhances system performance while maintaining data integrity.

The database schema includes tables for managing administrators, blood depots, donation campaigns, blood inventory, donation events, donors, reward balances, reward transactions, and user accounts. Each table is carefully structured with appropriate data types, constraints, primary keys, and foreign keys to maintain data integrity and facilitate efficient data retrieval, exemplifying the principles of normalization.

3.7 Interface Design

The system features three primary interfaces for user interaction. The main page serves as the gateway, offering links to the Admin and Blood Depot Admin modules and the User login module. Dynamic webpage creation is facilitated through a template file structure comprising heading.php, body.php, and footer.php. This approach isolates coding to functional pages, simplifying the process of propagating interface changes across modules.

Each module maintains its own distinct set of default PHP/CSS/JS and other template files. This segregation enables independent modification of individual modules without impacting others. Visual representations of the main user interface components are depicted in Figures 3.6 through 3.21.

Additionally, Bootstrap 4.5 was leveraged to craft the style elements of the initial landing webpage, ensuring a modern and responsive design.

The login interface is designed with simplicity, featuring user-friendly UI elements and calming colours to ensure minimal distraction for users. Within the WBMS, three primary login options are available: Admin, Blood Depot Admin and the User. After successful authentication, users are seamlessly redirected to the main page with access to the system's functionalities.

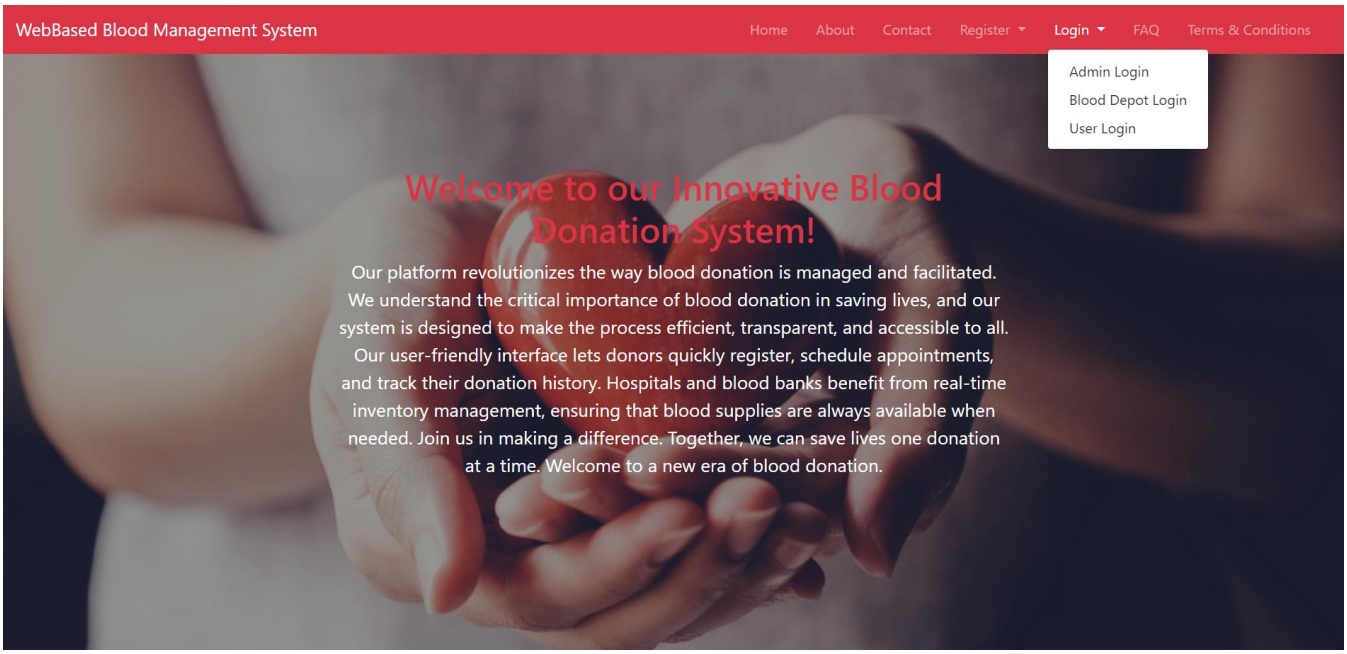


Figure 3.4 initial login screen

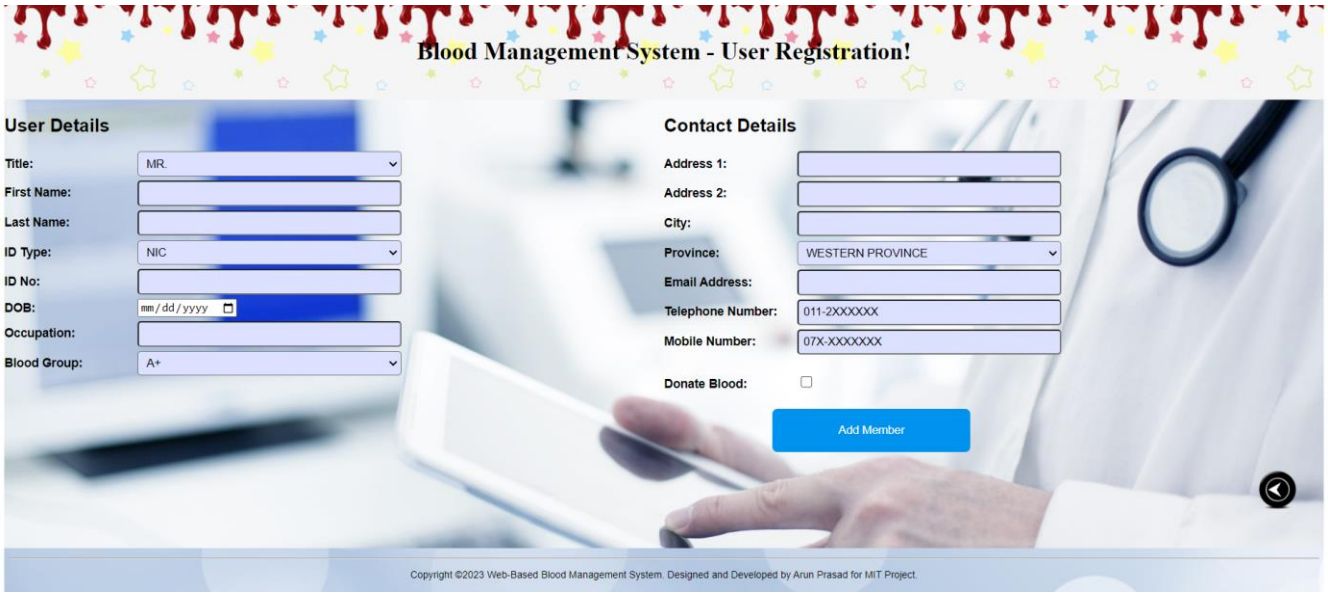


Figure 3.5 User registration interface.

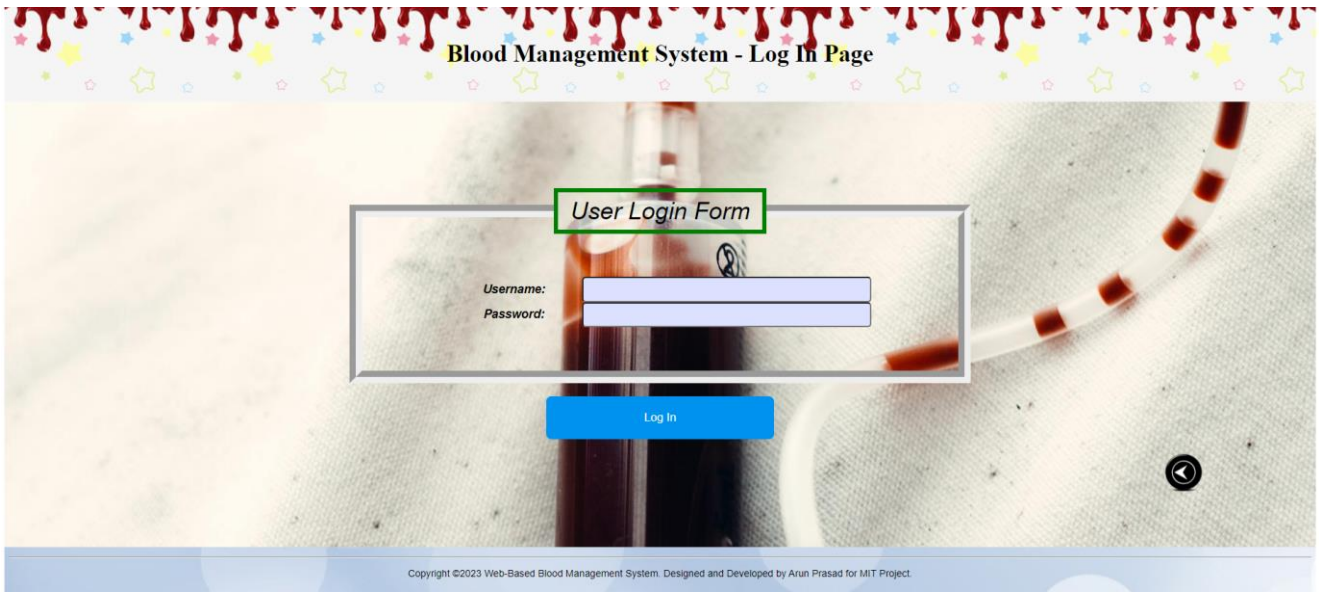


Figure 3.6 Admin User Login interface

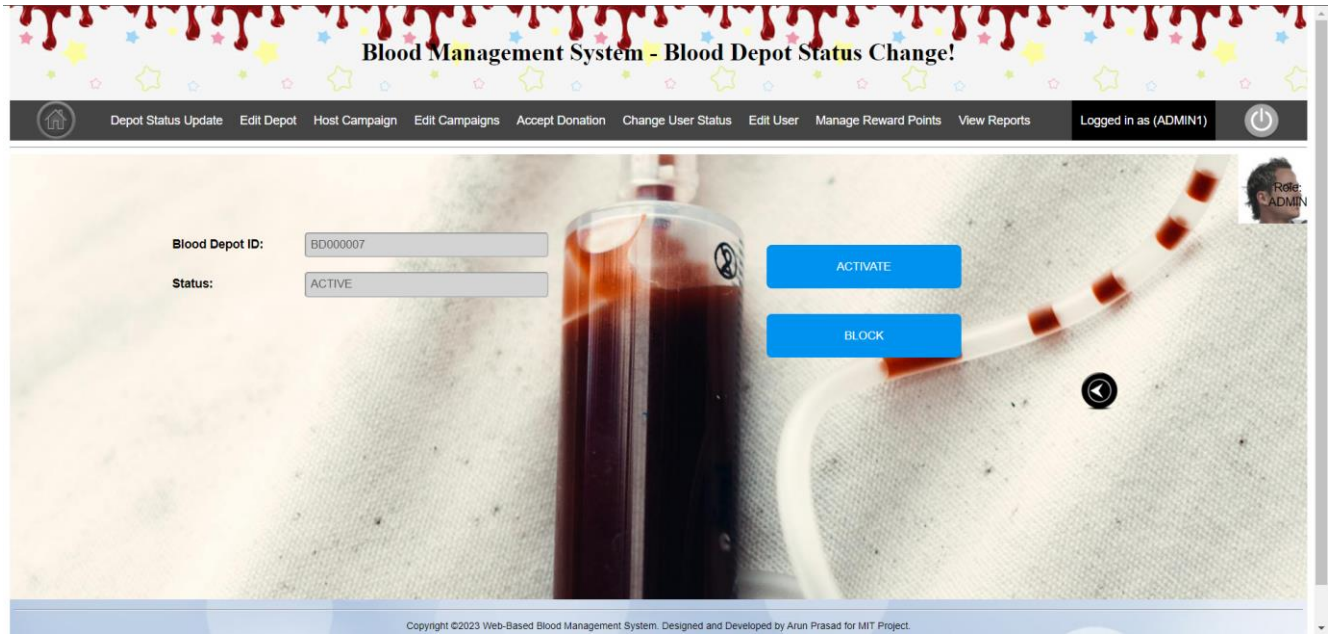


Figure 3.7 Blood Depot Activation Initial Screen

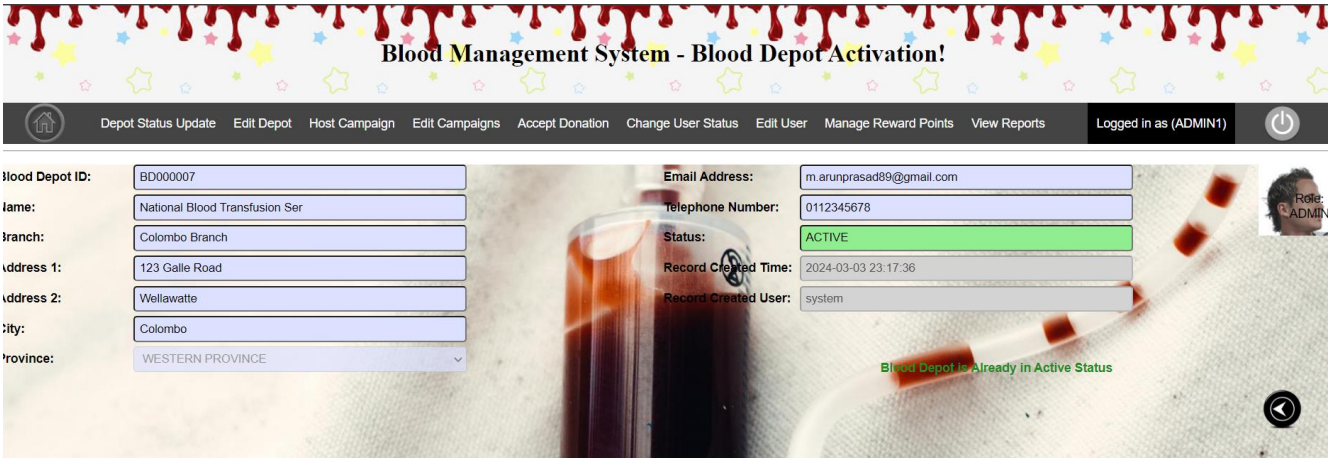


Figure 3.8 Blood Depot Activation Detail Screen

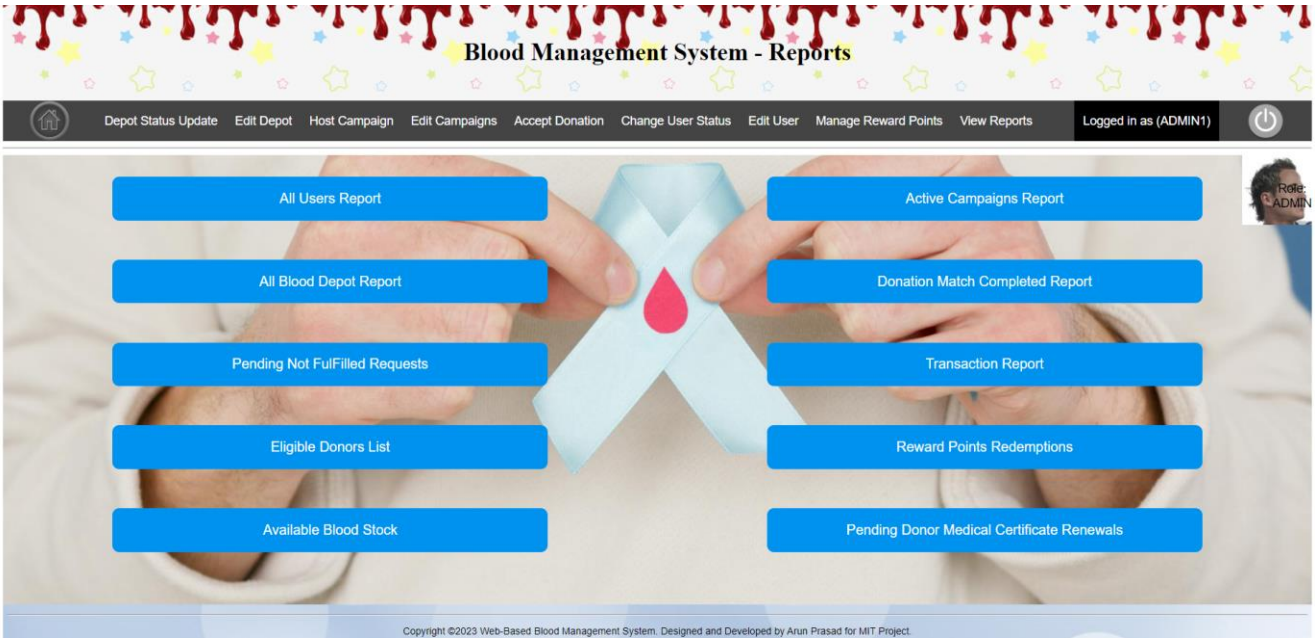
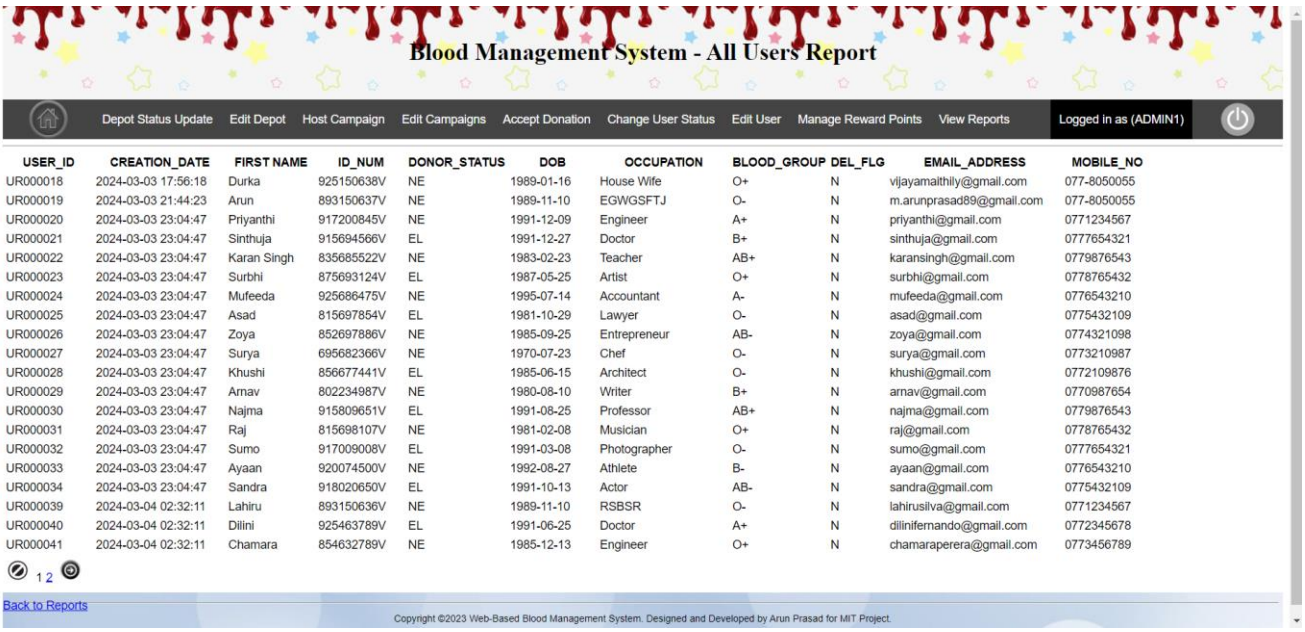


Figure 3.9 Management Report Interface



USER_ID	CREATION_DATE	FIRST NAME	ID_NUM	DONOR_STATUS	DOB	OCCUPATION	BLOOD_GROUP	DEL_FLG	EMAIL_ADDRESS	MOBILE_NO
UR000018	2024-03-03 17:56:18	Durka	925150638V	NE	1989-01-16	House Wife	O+	N	vijayamaithily@gmail.com	077-8050055
UR000019	2024-03-03 21:44:23	Arun	893150637V	NE	1989-11-10	EGWGSFTJ	O-	N	m.arunprasad89@gmail.com	077-8050055
UR000020	2024-03-03 23:04:47	Priyanthi	917200845V	NE	1991-12-09	Engineer	A+	N	priyanthi@gmail.com	0771234567
UR000021	2024-03-03 23:04:47	Sinthuja	915694566V	EL	1991-12-27	Doctor	B+	N	sinthuja@gmail.com	0777654321
UR000022	2024-03-03 23:04:47	Karan Singh	835685522V	NE	1983-02-23	Teacher	AB+	N	karansingh@gmail.com	0779876543
UR000023	2024-03-03 23:04:47	Surbhi	875693124V	EL	1987-05-25	Artist	O+	N	surbhi@gmail.com	0778765432
UR000024	2024-03-03 23:04:47	Mufeeda	925686475V	NE	1995-07-14	Accountant	A-	N	mufeeda@gmail.com	0776543210
UR000025	2024-03-03 23:04:47	Asad	815697854V	EL	1981-10-29	Lawyer	O-	N	asad@gmail.com	0775432109
UR000026	2024-03-03 23:04:47	Zoya	852697886V	NE	1985-09-25	Entrepreneur	AB-	N	zoya@gmail.com	0774321098
UR000027	2024-03-03 23:04:47	Surya	695682366V	NE	1970-07-23	Chef	O-	N	surya@gmail.com	0773210987
UR000028	2024-03-03 23:04:47	Khushi	856677441V	EL	1985-06-15	Architect	O-	N	khushi@gmail.com	0772109876
UR000029	2024-03-03 23:04:47	Arnav	802234987V	NE	1980-08-10	Writer	B+	N	arnav@gmail.com	0770987654
UR000030	2024-03-03 23:04:47	Najma	915809651V	EL	1991-08-25	Professor	AB+	N	najma@gmail.com	0779876543
UR000031	2024-03-03 23:04:47	Raj	815698107V	NE	1981-02-08	Musician	O+	N	raj@gmail.com	0778765432
UR000032	2024-03-03 23:04:47	Sumo	917009008V	EL	1991-03-08	Photographer	O-	N	sumo@gmail.com	0777654321
UR000033	2024-03-03 23:04:47	Ayaan	920074500V	NE	1992-08-27	Athlete	B-	N	ayaan@gmail.com	0776543210
UR000034	2024-03-03 23:04:47	Sandra	918020650V	EL	1991-10-13	Actor	AB-	N	sandra@gmail.com	0775432109
UR000039	2024-03-04 02:32:11	Lahiru	893150636V	NE	1989-11-10	RSBSR	O-	N	lahirusilva@gmail.com	0771234567
UR000040	2024-03-04 02:32:11	Dilini	925463789V	EL	1991-06-25	Doctor	A+	N	dilinifernando@gmail.com	0772345678
UR000041	2024-03-04 02:32:11	Chamara	854632789V	NE	1985-12-13	Engineer	O+	N	chamaraperera@gmail.com	0773456789

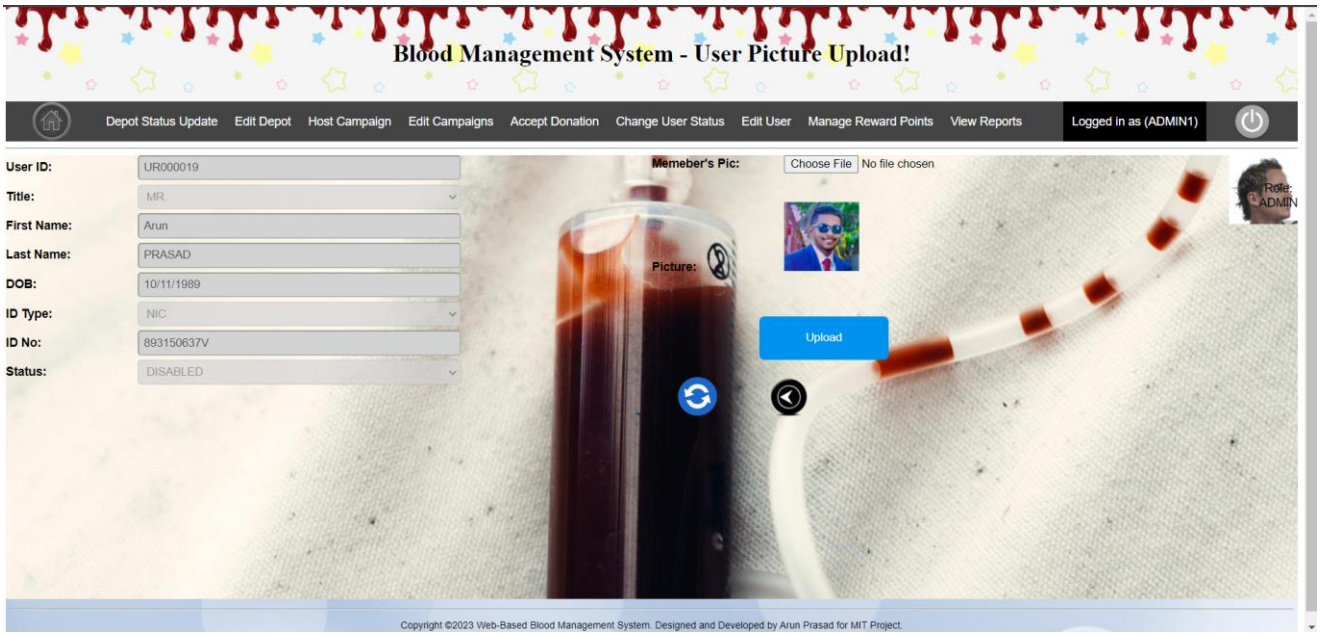


Figure 3.11 User picture upload

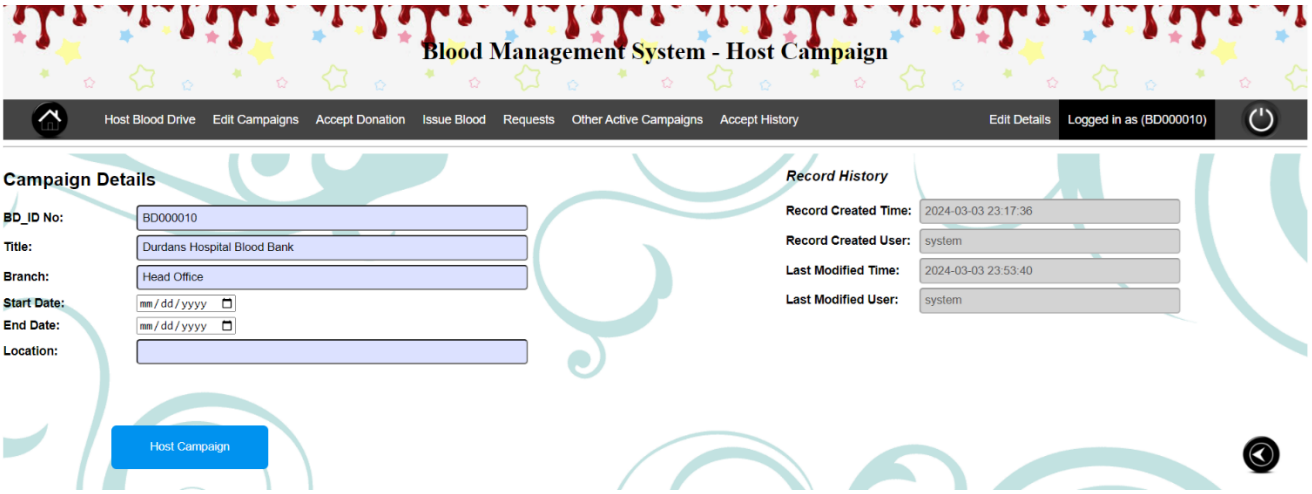


Figure 3.12 Blood Depot User Host Campaign Screen

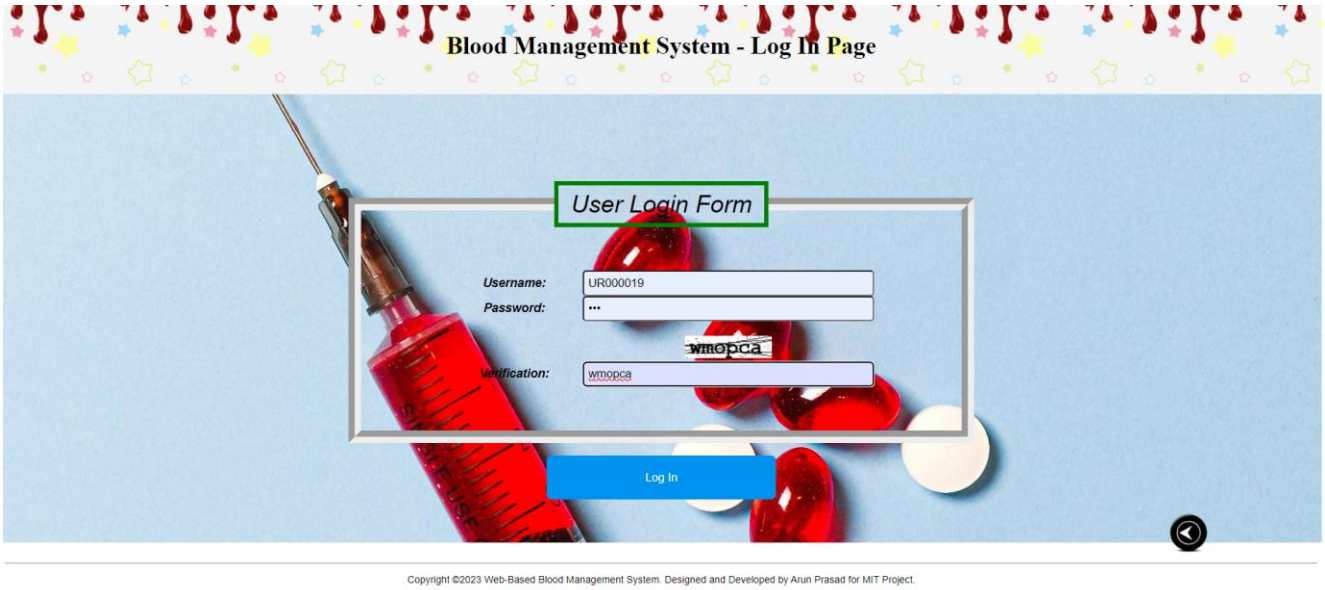


Figure 3.13 Donor/ Recipient User Login screen

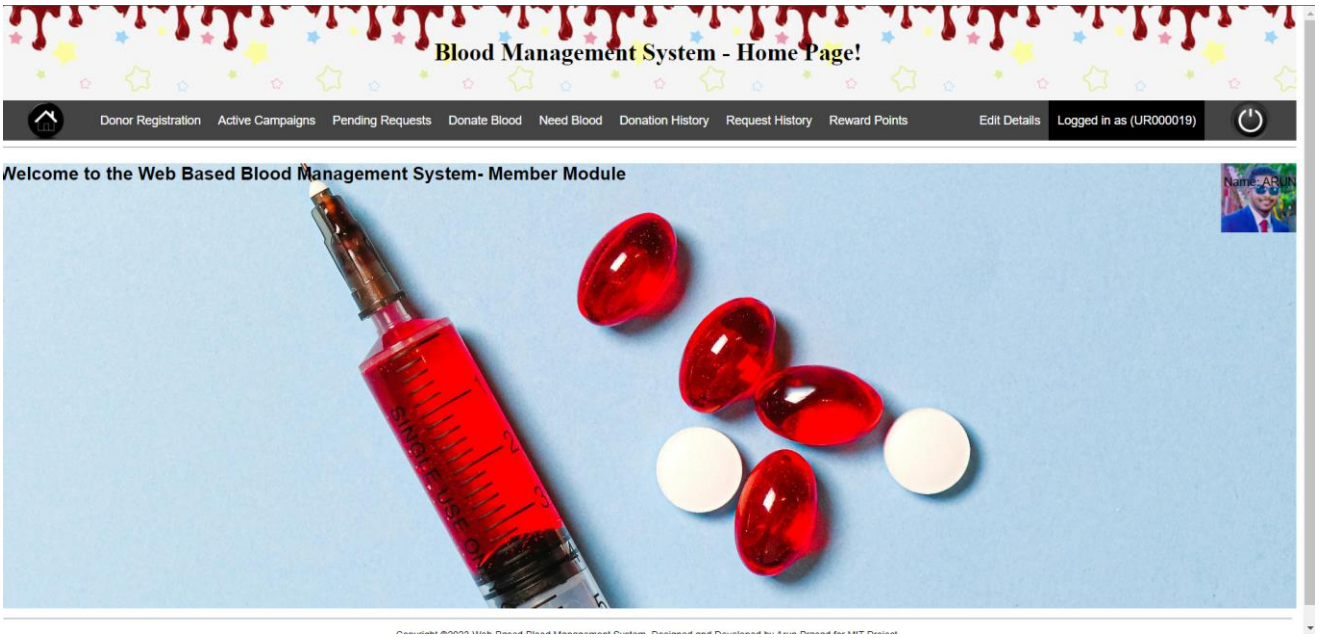
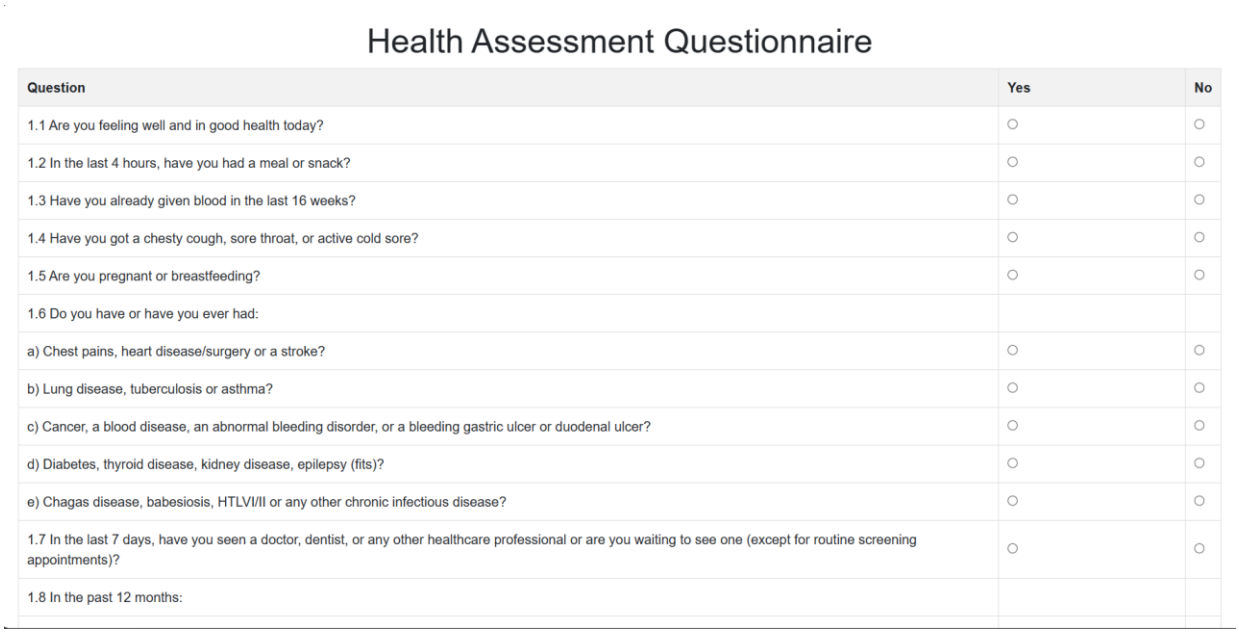


Figure 3.14 Donor/ Recipient User Home screen



Health Assessment Questionnaire

Question	Yes	No
1.1 Are you feeling well and in good health today?	○	○
1.2 In the last 4 hours, have you had a meal or snack?	○	○
1.3 Have you already given blood in the last 16 weeks?	○	○
1.4 Have you got a chesty cough, sore throat, or active cold sore?	○	○
1.5 Are you pregnant or breastfeeding?	○	○
1.6 Do you have or have you ever had:		
a) Chest pains, heart disease/surgery or a stroke?	○	○
b) Lung disease, tuberculosis or asthma?	○	○
c) Cancer, a blood disease, an abnormal bleeding disorder, or a bleeding gastric ulcer or duodenal ulcer?	○	○
d) Diabetes, thyroid disease, kidney disease, epilepsy (fits)?	○	○
e) Chagas disease, babesiosis, HTLVI/II or any other chronic infectious disease?	○	○
1.7 In the last 7 days, have you seen a doctor, dentist, or any other healthcare professional or are you waiting to see one (except for routine screening appointments)?	○	○
1.8 In the past 12 months:		

Figure 3.15 Health Assessment Questionnaire prior to Blood Donation

3.8 Summary

This chapter explained how to systematically design a system by considering its architecture, database design, and user interface, establishing the foundation for the implementation phase. The upcoming section of the document will cover the implementation of applications related to this WBMS.

Chapter 4 : IMPLEMENTATION

4.1 Introduction

Once the crucial analysis and design stages of the system development lifecycle are complete, the pivotal system implementation phase commences. This phase involves the actual system development based on the requirements and the chosen design for the project. Its primary goal is to transform the detailed design into a functional system, marking a significant milestone in the development process.

This chapter outlines the development tools employed in creating the web-based blood management system. It covers the hardware and software requirements, as well as the implementation environment. Additionally, it includes essential codes or reused code snippets, along with validations utilized in the system.

Furthermore, this chapter provides a comprehensive overview of the implemented system, including the technologies, programming languages employed, and database usage during development.

4.2 System Architecture

The application tier utilizes the PHP programming language to manage process-level business logic efficiently and effectively. The system data is stored in the MYSQL database layer. This approach ensures a well-organized and efficient system design that enables seamless data flow and processing.

In this web-based blood management system, users interact with the system through front-end webpages via their web browsers. When needed, the system communicates with the application server to access the database for data retrieval or modification.
Flow of interaction of the system is shown in below figure

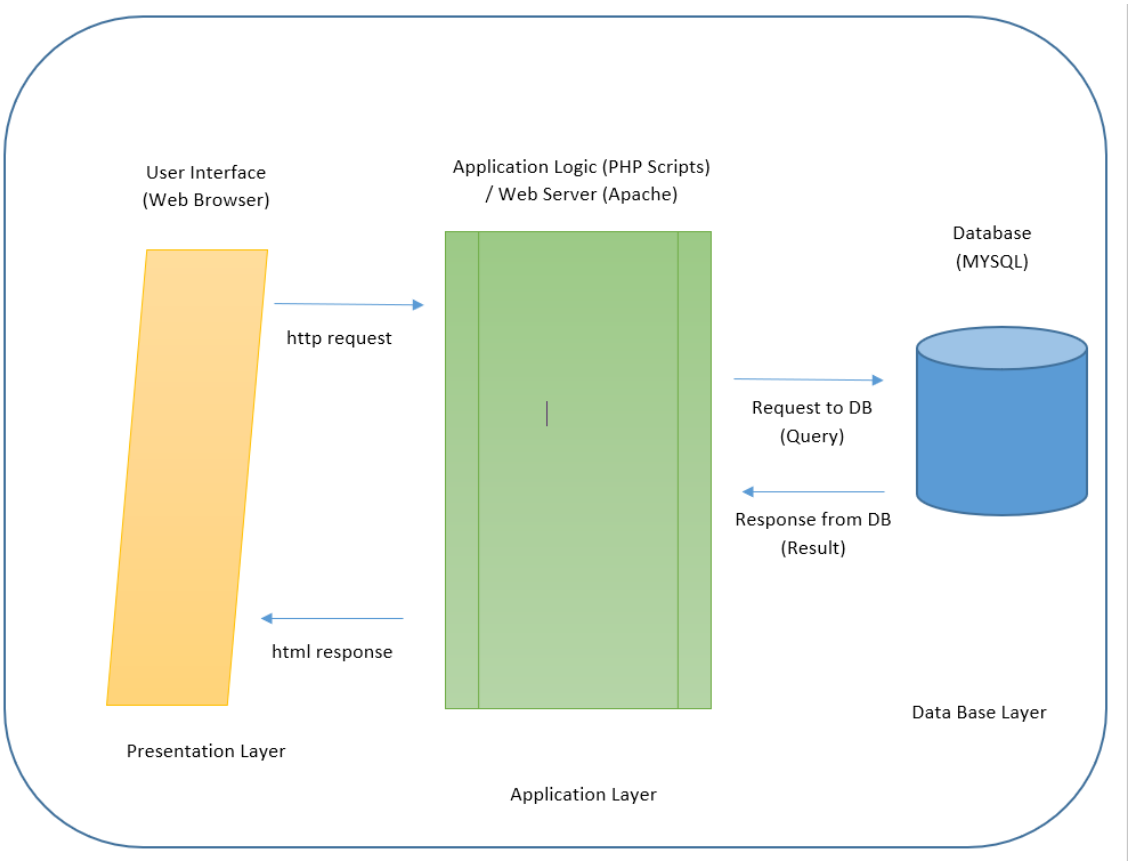


Figure 4.1 Flow of interaction of the system

4.3 Implementation Environment

4.3.1 Hardware Environment

- Core I7 Intel Processor 2.4 GHz (4 Cores) - 64 bit
- 8 GB RAM
- 500 GB SSD Hard Disk

Justification of choosing the Hardware Environment

- **Core i7 Intel Processor 2.4 GHz (64 bit):** The Core i7 processor is a powerful CPU suitable for handling resource-intensive tasks. Its 64-bit architecture allows for efficient processing of large datasets and complex computations, which benefits our system's performance.

- **8 GB RAM**: This amount of RAM is adequate for running multiple applications simultaneously without significant performance degradation. It ensures smooth execution of our system and supports efficient multitasking.
- **500 GB SSD Hard Disk**: The SSD (Solid State Drive) offers faster data access and improved overall system responsiveness compared to traditional HDDs. It enhances the system's speed, reduces loading times, and provides ample storage capacity for our project's requirements.

4.3.2 Software Environment

- PHP version 8.2.13
- Apache Web Server 2.4.58
- MYSQL database 8.2.0
- Window 11 Operating System 64 bit

Justification of choosing the Software Environment

- **PHP version 8.2.13**: PHP is a popular server-side scripting language known for its versatility and robustness in web development. Version 8.2.13 offers the latest features, improvements, and security enhancements, ensuring reliable and efficient code execution.
- **Apache Web Server 2.4.58**: Apache is a widely used open-source web server known for its stability, scalability, and performance. Version 2.4.58 includes bug fixes, security patches, and performance optimizations, making it a suitable choice for hosting web applications.
- **MySQL database 8.2.0**: MySQL is a powerful and reliable relational database management system. Version 8.2.0 offers advanced features, improved performance, and enhanced security, making it suitable for storing and managing our system's data efficiently.
- **WampServer Version 3.3.2**: WampServer provides an integrated development environment for web development, combining Apache, MySQL, and PHP in a single package. Version 3.3.2 ensures compatibility, stability, and ease of use, facilitating seamless development and testing of our system.

- **Windows 11 Operating System 64 bit**: Windows 11 provides a user-friendly interface and extensive compatibility with various software tools and technologies. Its 64-bit architecture allows for better utilization of system resources and supports modern software development environments.

4.4 Developments Tools and Used Technologies

When developing the system following tools and technologies are been used for the development.

- PHP
- MYSQL database
- Apache Web Server
- HTML
- CSS
- JavaScript
- MYSQL workbench
- phpMyAdmin
- Note Pad ++
- BootStrap
- WAMP server

PHP, MySQL database, Apache Web Server: These core technologies form the foundation of our system, enabling server-side scripting, database management, and web hosting capabilities.

HTML, CSS, JavaScript, and Bootstrap are front-end technologies that allow for creating interactive and visually appealing user interfaces, enhancing the user experience of our system.

MySQL Workbench and phpMyAdmin: These tools provide graphical user interfaces for managing MySQL databases facilitating database design, administration, and data manipulation tasks.

4.5 Key codes, validations, and reused snippets

4.5.1 Template for Dynamic Web Pages

The main template, shown in Figure 4.2, was crucial for developing the system's dynamic web pages. It served as the fundamental framework, providing structure, layout, and design elements to produce interactive, easy-to-use web content.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" type="text/css" href="css/main.css">

    <link rel="stylesheet" href="css/bootstrap.css" >
    <script src="js/jquery-3.5.1.slim.js" ></script>
    <script src="js/main.js"></script>
    <script src="js/bootstrap.js"></script>

    <title>WebBased Blood Management System</title>
  </head>

<body>

<?php
  // Show heading detail
  require_once('heading.php');

  // Show the navigation menu
  require_once('navmenu.php');

  // Show body common content
  require_once('body.php');

  // Insert the page footer
  require_once('footer.php');
?>
```

Figure 4.2 Dynamic web pages creation template

Throughout the application, a standardized template is utilized to create web pages. This consistent structure enables seamless application of changes across all pages within a given module. Upon successful login to a module, the user utilizes navmenu.php to navigate between various functionalities. Depending on the user and their role, navmenu.php dynamically generates webpage links to facilitate user navigation.

4..2 Authenticate user sessions

The code below (see Figure 4.3) is used to verify the user's credentials in the User Login module. Once the login is successful, the system creates a session and cookies for the user. This allows the user to navigate through authorized pages without any inconvenience.
Even after the session has ended, the application permits users to access restricted pages without having to re-enter their passwords until either the user logs out from the system or the cookies expire.

```php
// When User hasn't Login yet and submited for login
if (!isset($_SESSION['user_id'])) {
  if (isset($_POST['submit'])) {
    // Connect to the DB
    $dbc = mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME);

    // Grab the user-entered data during form submission
    $user_username = strtoupper(mysqli_real_escape_string($dbc, trim($_POST['username'])));
    $user_password = mysqli_real_escape_string($dbc, trim($_POST['password']));
    $pass_phrase = mysqli_real_escape_string($dbc, trim($_POST['verify']));

    if (sha1($pass_phrase)==$_SESSION['pass_phrase']){


    if (!empty($user_username) && !empty($user_password)) {
      // Check for the username and password in the DB
      $query = "SELECT a.user_id, b.first_name,a.image_name FROM user_login_tbl a,user_master_tbl b WHERE a.user_id=b.user_id and
                a.user_id = '$user_username' AND a.user_pwd = sha(CONCAT(UPPER('$user_username'),'$user_password'))and b.del_flg='N'";
      $results = mysqli_query($dbc, $query);


      if (mysqli_num_rows($results) == 1) {
        // The log-in is valid, set session vars and cookies and redirect to the default home page of the user
        $row = mysqli_fetch_array($results);
        $_SESSION['user_id'] = $row['user_id'];
        $_SESSION['first_name'] = $row['first_name'];
          //If User's pic available
          if (!empty($row['image_name'])) {
              $_SESSION['image_name'] = $row['image_name'];
              setcookie('image_name', $row['image_name'], time() + (60 * 60 * 24 * 45));    // expires in 45 days
          }
          else {
              $_SESSION['image_name'] = 'no_pic.jpg';
              setcookie('image_name', 'no_pic.jpg', time() + (60 * 60 * 24 * 45));    // expires in 45 days
          }
        setcookie('user_id', $row['user_id'], time() + (60 * 60 * 24 * 45));    // expires in 45 days
        setcookie('first_name', $row['first_name'], time() + (60 * 60 * 24 * 45));  // expires in 45 days
        $home_url = 'http://' . $_SERVER['HTTP_HOST'] . dirname($_SERVER['PHP_SELF']) . '/index.php';
        header('Location: ' . $home_url);
      }
```

Figure 4.3 Code used to authenticate User access to the application (Login page)

4.5.3 Usages of Cookies and Session

The code shown in Figure 4.4 is used to initiate a session in the application. This code is used on all restricted web pages. If the cookie variable is already present, the session variable is assigned to the cookie details. This login allows for persistence even after the session has ended.

```php
<?php
  session_start();

  // If user_id Session variable is not set but cookie is set, then set session variables with cookie user_id/first_name info
  if (!isset($_SESSION['user_id'])) {
    if (isset($_COOKIE['user_id']) && isset($_COOKIE['first_name'])&&  isset($_COOKIE['image_name'])) {
      $_SESSION['user_id'] = $_COOKIE['user_id'];
      $_SESSION['first_name'] = $_COOKIE['first_name'];
      $_SESSION['image_name'] = $_COOKIE['image_name'];
    }
  }
?>
```

Figure 4.4 Session start code

4.5.4 Log out Script with destroying the stored Session and Cookies

The following script (See Figure 4.5) can be used to log out a user permanently by clearing session variables and removing stored cookies.

```php
<?php
  // If the user is logged in, delete the session vars, session cookie, destroy session and clear cookies
  session_start();
  if (isset($_SESSION['user_id'])) {
    // Delete the session vars
    $_SESSION = array();

    // Delete the session cookie
    if (isset($_COOKIE[session_name()])) {
      setcookie(session_name(), '', time() - 3600);
    }

    // Destroy the session
    session_destroy();
  }

  // Delete the membership_id , first_name,  mem_expiry_date and image_name cookies
  setcookie('user_id', '', time() - 3600);
  setcookie('first_name', '', time() - 3600);
  setcookie('image_name', '', time() - 3600);


  // Redirect to the home page
        $home_url = 'http://' . $_SERVER['HTTP_HOST'] . '/blood_management/index.php';
        header('Location: ' . $home_url);
?>
```

Figure 4.5 Logout script

4.5.5 Prevent from SQL Injections

To ensure the system is protected against SQL injection, the script utilizes mysqli_escape_string and trim functions, as illustrated in Figure 4.6. The script consistently follows the same approach to retrieve data from the $_POST method.

```
    if (isset($_POST['submit'])) {
            $dbc = mysqli_connect(DB_HOST, DB_USER, DB_PASSWORD, DB_NAME)  or die("Error connecting to MYSQL Database : ". mysqli_connect_error());

// Grab the POST data
$title = mysqli_real_escape_string($dbc, trim($_POST['TITLE']));
$first_name = strtoupper(mysqli_real_escape_string($dbc, trim($_POST['FIRST_NAME'])));
$last_name = strtoupper(mysqli_real_escape_string($dbc, trim($_POST['LAST_NAME'])));
$id_type = mysqli_real_escape_string($dbc, trim($_POST['ID_TYPE']));
$id_num = strtoupper(mysqli_real_escape_string($dbc, trim($_POST['ID_NUM'])));
$dob = mysqli_real_escape_string($dbc, trim($_POST['DOB']));
$blood_group = mysqli_real_escape_string($dbc, trim($_POST['BLOOD_GROUP']));
$address1 = strtoupper(mysqli_real_escape_string($dbc, trim($_POST['ADDRESS1'])));
$address2 = strtoupper(mysqli_real_escape_string($dbc, trim($_POST['ADDRESS2'])));
$city = strtoupper(mysqli_real_escape_string($dbc, trim($_POST['CITY'])));
$province = mysqli_real_escape_string($dbc, trim($_POST['PROVINCE']));
$occupation = strtoupper(mysqli_real_escape_string($dbc, trim($_POST['OCCUPATION'])));
$email_address = mysqli_real_escape_string($dbc, trim($_POST['EMAIL_ADDRESS']));
$tp_no = mysqli_real_escape_string($dbc, trim($_POST['TP_NO']));
$mobile_no = mysqli_real_escape_string($dbc, trim($_POST['MOBILE_NO']));


//if checked POST variable is not empty, Thus Value will be set to Y else N
if (!empty($_POST['DONOR'])){
$donor=mysqli_real_escape_string($dbc, trim($_POST['DONOR']));
}
else{
$donor="N";
}
```

```
// Check whetеher all mandatory data keyed in
if(!empty($title) && !empty($first_name) && !empty($last_name) && !empty($id_type) && !empty($id_num) && !empty($dob) &&
    !empty($address1) && !empty($address2) && !empty($city) && !empty($province) && !empty($occupation) &&
     !empty($email_address) && !empty($tp_no)&& !empty($mobile_no))
    {
          require_once('validation.php');

          //Field Validations- NIC, DATE, PHONE NO, EMAIL
          if (email_validation($email_address)&& phone_validation($tp_no)&& phone_validation($mobile_no)
             &&  date_validation($dob) && NIC_validation($id_type,$id_num)){

//insert statements begins here
    begin($dbc);
// insert to the user_master_tbl
$query = "INSERT INTO user_master_tbl(TITLE, FIRST_NAME, LAST_NAME, ID_TYPE, ID_NUM, DOB,DONOR_FLG,DONOR_STATUS, BLOOD_GROUP,  ADDRESS1, ADDRESS2, CITY, PROVINCE, OCCUPATION,
            EMAIL_ADDRESS, TP_NO, MOBILE_NO, R_CRE_TIME, R_CRE_USER_ID, L_CHG_TIME, L_CHG_USER_ID) values (
            '$title' ,'$first_name' ,'$last_name' ,'$id_type' ,'$id_num' ,STR_TO_DATE('$dob','%Y-%m-%d'),'$donor', '$blood_group' ,'NE' ,'$address1' ,'$address2' ,'$city' ,
            '$province' , '$occupation' ,
            '$email_address', '$tp_no', '$mobile_no', now(), 'SYSTEM', now(), 'SYSTEM')";

 $result=mysqli_query($dbc, $query);

if($result){

          //get newly createduser id
$query1 = "SELECT USER_ID from user_master_tbl WHERE id_num='$id_num' and del_flg='N'";

 $result1=mysqli_query($dbc, $query1);
  while ($row = mysqli_fetch_array($result1)){
     $user_id=$row['USER_ID'];  }

//Write into Login Table
if (!empty($user_id)){
    $query1 = "INSERT INTO user_login_tbl (USER_ID, USER_PWD, NO_OF_ATTEMPTS, R_CRE_TIME, R_CRE_USER_ID)
            values ('$user_id', sha(CONCAT(UPPER('$user_id'),'$password')), '0',now(), 'SYSTEM')";

$result1=mysqli_query($dbc, $query1);

    if($result1){
        commit($dbc);

      $err_msg="Member Successfully Activated. Login Password sent to $email_address";
```

Figure 4.6 SQL injection prevention techniques

After a user submits a record, the script retrieves the submitted data and inserts the details into a MYSQL database.

4.5.6 Core Reusable Functions

Reusable functions are deployed to confirm blood group compatibility between donors and requestors, aiding in successfully matching donors with needy recipients. These reusable functions help identify eligible donors and ultimately enhance the likelihood of successful blood transfusions.

The illustrated reusable function (refer to Figure 4.7) to ascertain the compatibility of donors' blood groups with a given requestor's blood group.

```php
function getRequesterCompatibleBloodGroups($bloodGroup) {
    $compatibleGroups = array();

    switch ($bloodGroup) {
        case 'A+':
            $compatibleGroups = array('A+', 'A-', 'O+', 'O-');
            break;
        case 'A-':
            $compatibleGroups = array('A-', 'O-');
            break;
        case 'B+':
            $compatibleGroups = array('B+', 'B-', 'O+', 'O-');
            break;
        case 'B-':
            $compatibleGroups = array('B-', 'O-');
            break;
        case 'AB+':
            $compatibleGroups = array('A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-');
            break;
        case 'AB-':
            $compatibleGroups = array('A-', 'B-', 'AB-', 'O-');
            break;
        case 'O+':
            $compatibleGroups = array('O+', 'O-');
            break;
        case 'O-':
            $compatibleGroups = array('O-');
            break;
        default:
            $compatibleGroups = array('Invalid blood group');
            break;
    }

    return $compatibleGroups;
}

$requesterBloodGroup = 'O+';
$requesterCompatibleGroups = getRequesterCompatibleBloodGroups($requesterBloodGroup);
echo "($requesterBloodGroup): can receive from" . implode(', ', $requesterCompatibleGroups) . "\n";
```

Figure 4.7 getRequesterBloodGroup function

The displayed reusable function (refer to Figure 4.8) determines the compatibility of a requestor's blood group with a given donor's.

```php
function getDonorCompatibleBloodGroups($bloodGroup) {
    $compatibleGroups = array();

    switch ($bloodGroup) {
        case 'A+':
            $compatibleGroups = array('A+', 'AB+');
            break;
        case 'A-':
            $compatibleGroups = array('A+', 'A-', 'AB+', 'AB-');
            break;
        case 'B+':
            $compatibleGroups = array('B+', 'AB+');
            break;
        case 'B-':
            $compatibleGroups = array('B+', 'B-', 'AB+', 'AB-');
            break;
        case 'AB+':
            $compatibleGroups = array('AB+');
            break;
        case 'AB-':
            $compatibleGroups = array('AB+', 'AB-');
            break;
        case 'O+':
            $compatibleGroups = array('A+', 'B+', 'O+', 'AB+');
            break;
        case 'O-':
            $compatibleGroups = array('A+', 'A-', 'B+', 'B-', 'O+', 'O-', 'AB+', 'AB-');
            break;
        default:
            $compatibleGroups = array('Invalid blood group');
            break;
    }

    return $compatibleGroups;
}
```

Figure 4.8 getRequesterDonor function

Within PHP scripts, the "begin," "commit," and "rollback" functions are used to simplify coding and enable their seamless integration across all scripts. These functions are handy when executing multiple SQL statements within a PHP script for a specific event, such as updating a database.

When executed, the "begin" function initiates a transaction, which allows for the execution of multiple SQL statements as one unit of work. If all expected SQL statements in the script execute successfully, the "commit" function is called to finalize and commit the changes to the database.

However, if any of the SQL statements fail to execute, the "rollback" function is called. This function rolls back all changes made to the database during the transaction, ensuring the database remains in its original state.

These functions streamline coding, make it easier to implement across all PHP scripts and ensure that database transactions are executed reliably and consistently.

```php
<?php

// trans.php
function begin($dbc){
    mysqli_query($dbc,"BEGIN");
}

function commit($dbc){
    mysqli_query($dbc,"COMMIT");
}

function rollback($dbc){
    mysqli_query($dbc,"ROLLBACK");
}

// image max file size
$max_size=131072;

if(isset($_SESSION['user_id'])){
 $user=strtoupper($_SESSION['user_id']);
 $fname=strtoupper($_SESSION['first_name']) ;
 $image_name=$_SESSION['image_name'] ;

?>

<!-- User Pic in Right hand side Top Corner-->
    <div class="r_corner">
    <tr><td><img style="float: right;" src='<?php  if (!empty($image_name)) echo 'images/user/'.$image_name ; ?>'
    alt="Member Profile Picture" /></td></tr>
    <br/>
        <p style="float: right;">Name: <?php echo $fname ; ?></p>
    </div>
```

Figure 4.9 User module body.php

Additionally, The HTML code shown in Figure 4.9 ensures that once a user successfully logs into the Blood Donor/Recipient module, their profile picture will be shown in the top-right corner of the webpage.

4.5.7 Java Script Validation

The JavaScript code shown in Figure 4.10 was used to validate the length and values of NICs. Similar JavaScript code was also applied for front-end data validations when registering users and Blood Depots into the system and other modules where data is inserted. These validations included checking for non-empty fields and confirming the date, phone number, and email formats using similar JavaScript codes.

```
function validateNIC(inputField, helpText) {


   if ((inputField.value.length != 10) && (inputField.value.length != 12)) {
     if (helpText != null){
       helpText.innerHTML = "NIC should Contain 10 or 12 chars.";}
     return false;
   }


   if (inputField.value.length ==12) {
     if (isNaN(inputField.value)){
       if (helpText != null){
       helpText.innerHTML = "12 Digits NIC should only contain Numbers";}
     return false;
     }
   }

   if (inputField.value.length ==10) {
     if (isNaN(inputField.value.substring(0, 9))){
       if (helpText != null){
       helpText.innerHTML = "First 9 Digits should only contain Numbers";}
     return false;
     }
   }

   if (inputField.value.length ==10) {
     if ((inputField.value.substring(9, 10) != 'V') && (inputField.value.substring(9, 10) != 'X')){
       if (helpText != null){
       helpText.innerHTML = "Last Digit of NIC should be V or X";}
     return false;
          }

       else {
       if (helpText != null)
       helpText.innerHTML = "";
       return true;
       }
   }
   else {
       if (helpText != null)
       helpText.innerHTML = "";
       return true;
```

```
 function validateNumber(inputField, helpText) {
   if (!validateNonEmpty(inputField, helpText))
       return false;
   if (isNaN(inputField.value)){
     if (helpText != null)
       helpText.innerHTML = "Please enter a Number.";
     return false;
   }
    else {
     // Validation Passed. So clear the help msg in span tag
     if (helpText != null)
       helpText.innerHTML = "";
     return true;
   }
 }
```

Figure 4.10 JavaScript validations

4.5.8 PHP Validation

JavaScript validations display validation errors to users as they navigate away from a specific field. However, the primary validation is done within PHP scripts as the second and final line of defence (Figure 4.11).

The system validates the NIC, email, phone number pattern, and date format when a form is submitted using the custom functions mentioned below. Form data submission is only allowed if the validation is successful.

```
    function date_validation($date){
            if (!preg_match('/^(\d{4})-\d{2}-\d{2}$/',$date)){
                return false;
        }
        else {
            return true;
        }
        }
    function NIC_validation($id_type,$id_num){
        //check whether id type is NIC or DL
            if ($id_type=="NIC"||$id_type=="DL" ){
            //If no of digit is 10
    if (strlen($id_num) == 10) {
            $id_num_upper=strtoupper($id_num);
                if (is_numeric(substr($id_num,0,9)) && ((substr($id_num_upper,9,1) == 'V') || (substr($id_num_upper,9,1) =='X')))
                                                    {
            return true;
                                                    }
        else {
            return false;
             }
                                    }
                //If no of digit is 12
    else if (strlen($id_num) == 12) {
        if (is_numeric($id_num)){
            return true;
                    }
        else {
            return false;
                }
            }
    else {
        return false;
         }

            }
            //ID TYPE IS PP so no Validation Required
            else {
                return true;
                }
    }

//echo NIC_validation("NIC","893150617V");
```

Figure 4.11 Date, NIC validation - PHP function

Following regular expressions are used to validate email, phone number pattern and date fields in PHP scripts which is shown in figure 4.12

```php
<?php
    function email_validation($email){
        if (!preg_match('/^[a-zA-Z0-9][a-zA-Z0-9?\.!=\-_&]*@[\w-]+(\.\w{2,3})+$/',$email)){
            return false;
        }
        else {
            return true;
        }
        }
    function phone_validation($tp_no){
        if (!preg_match('/^\d{3}-\d{7}$/',$tp_no)){
            return false;
        }
        else {
            return true;
        }
        }
```

Figure 4.12 Email, Phone validations - PHP functions

4.5.9 Data Base Triggers

In light of MySQL's lack of sequence support, separate sequence tables were adopted to manage the subsequent sequence number for users. A database-level trigger was introduced to precede record insertions into either the ‘user_master_tbl’ or ‘blood_depot_master_tbl’ to meet this need. Consequently, user IDs are assigned a 'UR' prefix in the ‘user_master_tbl’, while Blood Depot IDs are assigned a 'BD' prefix in the ‘blood_depot_master_tbl’.

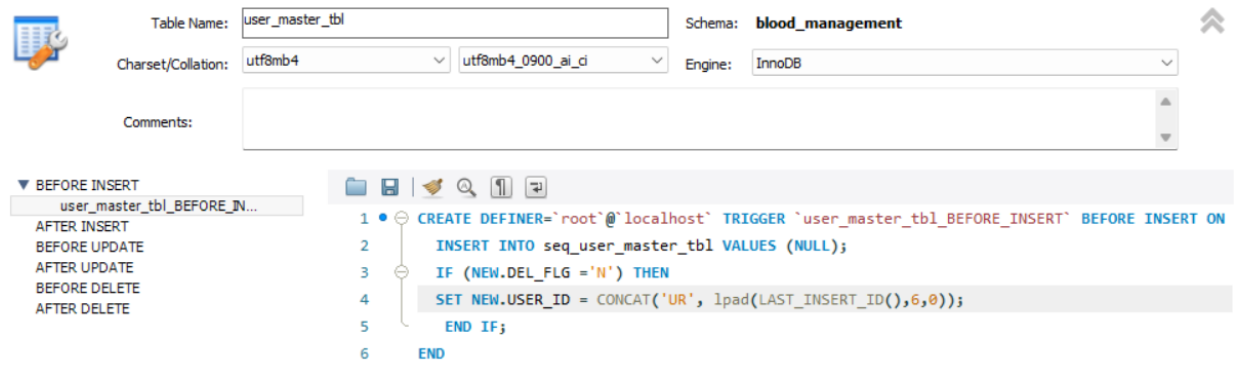


Figure 4.13 Database Trigger (User_Master_Tbl)

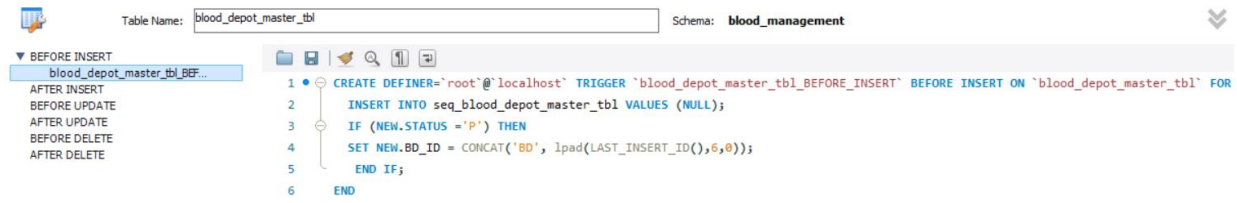


Figure 4.14 Database Trigger (blood_depot_master_tbl)

4.5.10 Other Reused Codes

• Utilized the PHP mail function to send email reminders to clients. Given the system's operation within a Windows environment, the sendmail tool acquired from http://glob.com.au/sendmail/ was incorporated into the Wamp64 directory to interface with PHP. Configuration included setting up an SMTP connection to facilitate the sending of email reminders to users.

sendmail.exe is a simple windows console application that emulates sendmail's -t option to deliver emails piped via stdin. it is intended to ease running unix code that has /usr/lib/sendmail hardcoded as an email delivery means. it doesn't support deferred delivery, and requires an smtp server to perform the actual delivery of the messages. generally all you need to do is install sendmail.exe in \usr\lib, and existing code that calls /usr/lib/sendmail will work. if you're coding new applications, all you need to do is construct your email message with complete headers, then pipe it to /usr/lib/sendmail -t(“fake sendmail for windows,” n.d.)

• Styled the initial home page of the web-based system using Bootstrap 4.5.

4.6 Deployment Summary

User Interface	Html, CSS, Bootstrap
Backend	PHP
Database	MY SQL
Hosted server	WAMP

Table 4.1 Deployment Summary

4.7 Summary

This Chapter covered the architecture, hardware and software aspects, development tools and technologies, critical codes, validations, and reused snippets essential for the project's functionality. The Chapter also discusses measures to prevent SQL injections, core reusable functions, JavaScript and PHP validation techniques, and database triggers. Finally, it concludes with a deployment summary summarizing the key points and processes of deploying the implemented system.

Chapter 5 : EVALUATION

5.1 Introduction

The Testing and Evaluation chapter of the software profiling report aims to assess the application's usability and functionality through manual and automated testing approaches. The automated testing framework utilizes BDD (Behavior-Driven Development) with Cucumber and Selenium WebDriver, integrated with IntelliJ IDEA IDE, for test automation.

Testing is one of the critical areas of the system development lifecycle, as it helps to identify the system's grey areas and clear them out so that the developed system is more accurate and meets the user requirements and expectations.

5.2 Testing Objectives

The primary objectives of testing are as follows:

- Evaluate the usability and functionality of the software.
- Identify any defects or issues in the application.
- Validate the software against specified requirements.
- Measure the effectiveness of automated testing using BDD with Cucumber and Selenium WebDriver in IntelliJ IDEA.

5.2.1 Manual Testing:

- Exploratory Testing: Explore the application's features and functionalities without predefined test cases to uncover potential issues.
- Functional Testing: Verify that each software function operates according to the requirement specification.
- Usability Testing: Assess the software's ease of use, user interface design, and overall user experience.
- Regression Testing: Ensure new changes or updates do not adversely affect existing functionalities.

5.2.2 Automated Testing (BDD Cucumber with Selenium WebDriver):

- Develop automated test scenarios using the BDD approach with Cucumber to align with the software's business requirements.
- Utilize Selenium WebDriver for automated UI testing to simulate user interactions and validate application behaviour.
- Integrate automated tests with IntelliJ IDEA IDE for writing, executing, and managing test scripts.

5.3 Testing Procedures

5.3.1 Manual Testing Procedures:

- Create comprehensive test cases based on functional requirements, user stories, and usability.
- Execute manual tests and record observations, defects, and usability feedback.
- Conduct Regression testing after each software update to ensure backward compatibility.

5.3.2 Automated Testing Procedures:

- Develop feature files in BDD format using Cucumber's Gherkin syntax to define test scenarios.
- Implement step definitions in Java for automated test execution.
- Utilize Selenium WebDriver to automate UI interactions such as clicks, input validations, and form submissions.
- Execute automated tests within IntelliJ IDEA IDE using Cucumber runners and analyze test results.

5.4 Usability Report

5.4.1 Usability Testing Results: Feedback from Usability Testing Sessions:

Overall, users found the system intuitive and easy to navigate, appreciating the clear labelling and organization of features. However, they also highlighted the need for more prominent notifications and alerts for critical actions to enhance usability further.

5.4.2 Evaluation of User Interface Design:

The user interface design received positive feedback for its modern look and feel. Users also noted the effectiveness of the visual hierarchy and colour schemes in guiding their attention. Some suggestions were made for minor font size and contrast adjustments to improve readability.

5.4.3 Overall User Experience (UX) Evaluation:

Users generally had a positive experience using the software, although improvements in navigation, accessibility, and data entry processes were suggested to enhance their experience further.

5.4.4 Automated Testing Evaluation Effectiveness of Automated Testing with BDD

Automated testing with BDD using Cucumber and Selenium WebDriver proved highly effective in validating critical functionalities, thanks to well-defined test scenarios that facilitated collaboration between testers and developers.

5.4.4.1 Measurement of Automated Testing:

Test Coverage: Automated tests covered a significant portion of the system's functionalities, including complex workflows and edge cases.

Execution Time: Automated test execution time was notably faster than manual testing, leading to quicker feedback loops.

Reliability: Automated tests consistently produce reliable results, reducing the chances of human error.

Maintainability: Test scripts were modular and easy to maintain, allowing seamless updates as the system evolved.

5.4.4.2 Comparison with Manual Testing:

Efficiency: Automated testing significantly outperformed manual testing in speed and efficiency, especially for repetitive test cases.

Accuracy: Automated tests provided consistent and accurate results, minimizing the risk of overlooking critical issues.

Regression Testing Coverage: Automated testing enabled comprehensive regression testing, ensuring that new updates did not introduce unexpected bugs.

Overall, incorporating automated testing using BDD with Cucumber and Selenium WebDriver has dramatically enhanced the testing process, improving efficiency, accuracy, and regression testing coverage compared to manual testing methods.

5.5 Test Cases / Results

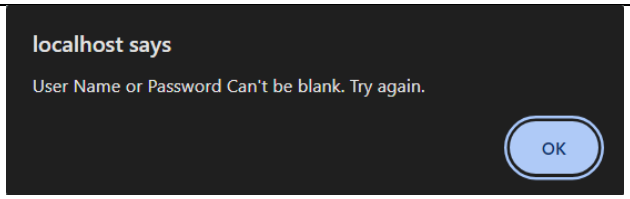
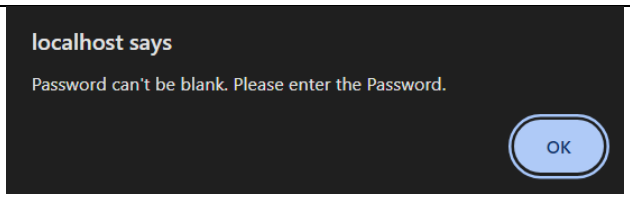
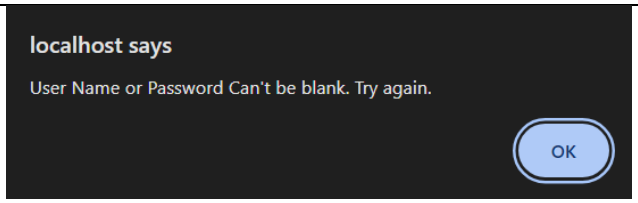
Test cases contain descriptions of test cases, their expected behaviour, and how the system responds to the case. In order to find the errors in the initial stages and improve the system's functionality, every module-wise test case has been created.

Here is a list of sample test cases and test results in the major modules.

5.5.1 Login Validations for Donor/Requestor User Logins

In the user login page ("Login.php"), when the correct user name and password are keyed in, the user is directed to index.php (After creating the user session and storing cookie details)

Table 5.2 below depicts the test results of the user login module.

No	Test Description	Result
1	When user tries to login without username and the password.	
2	When the user tries to login without password	
3	When the user tries to login without username	

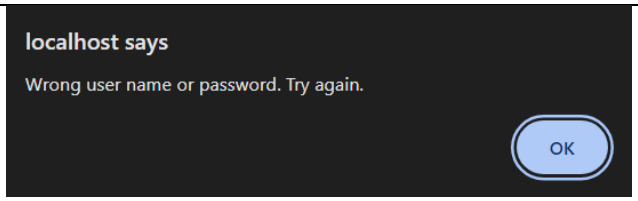
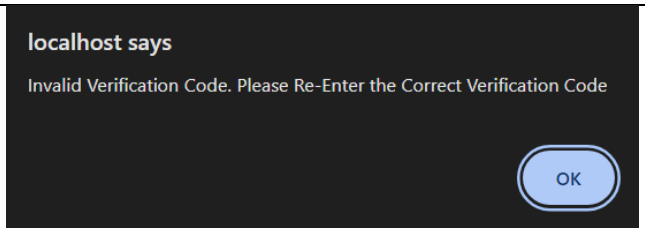
4	When the user tries to login with erroneous password	
5	When Entered Captcha Code is incorrect	

Table 5.1 Test cases/results for user login validation

5.5.2 Register User Validations and Test Cases/Results

Below mentioned table 5.3 depicts the test results of add user module.

When user keys in all the mandatory data, and submits the user record, it is being added to the user_master_tbl and system returns the appropriate User id to the user and the next sequence number this user id is being created.

No	Test Description	Result
1	When user moves away from a mandatory field without keying in the value.	
2	When user moves away from date field without keying in the date format as dd-mm-yyyy	
3	When user tries to move away from field with incorrect telephone no /mobile no format.	

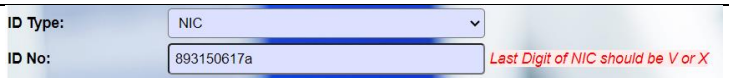
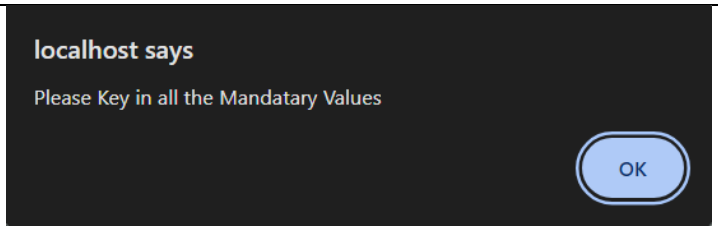
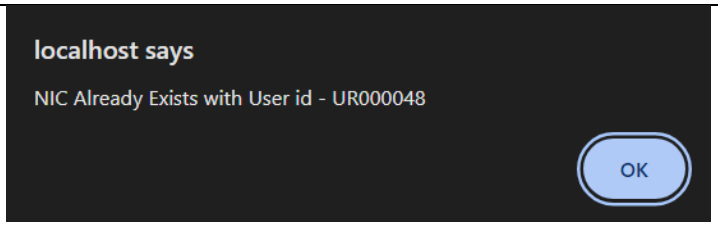
4	When user tabs out from NIC field without keying in proper NIC no. (NIC no length not equal to 10 or 12 digits)	
5	When user tabs out from NIC field without keying in proper NIC no. (incorrect last digit)	
6	When user tabs out from NIC field without keying in proper NIC no. (invalid char for 12-digit NIC)	
9	When user tries to submit the record without completing	
10	When an already registered user details being tried to add again. (duplicated id number)	

Table 5.2 Test cases/results for add user module validation

5.5.3 Automated Selenium Testing for Blood Depot Status Changes with BDD

The combination of Selenium WebDriver and Behavior-Driven Development (BDD) methodologies has been highly effective in the field of automated testing. This section delves into the specifics of utilizing automated Selenium testing for monitoring and managing blood depot status changes, employing BDD frameworks to enhance the testing process.

Sample of the BDD Feature File

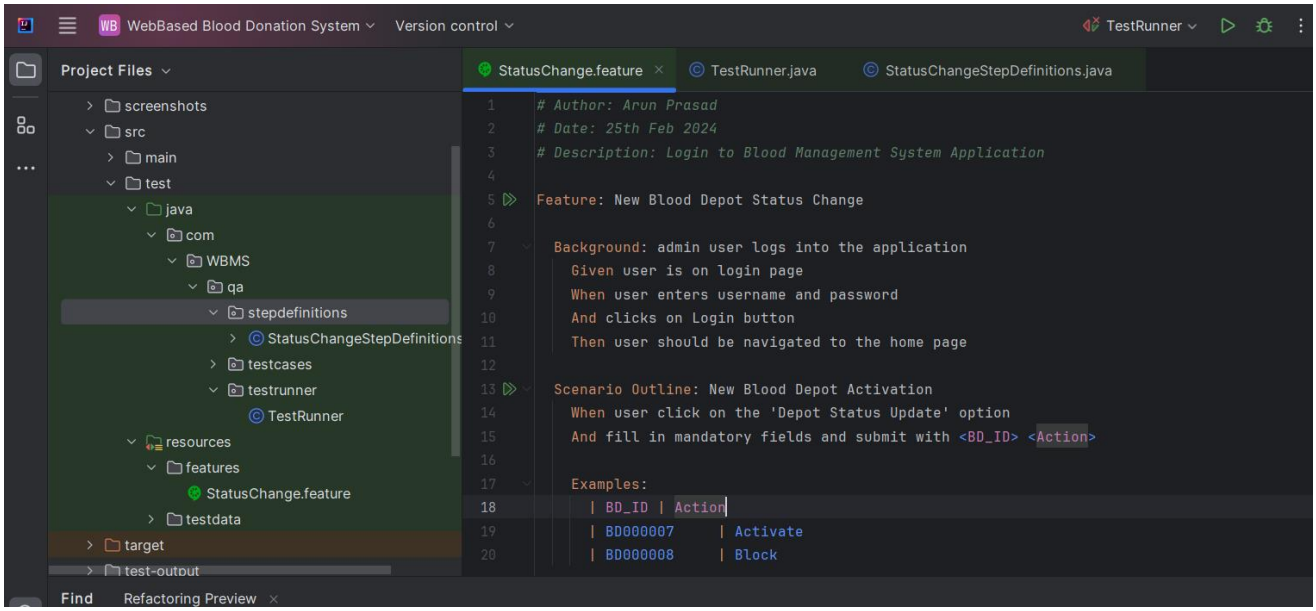


Figure 5.1 Sample BDD Feature File

Figure 5.1 showcases a sample BDD Feature File tailored for the automated testing of blood depot status changes. This file encapsulates scenarios and corresponding steps in a human-readable format, aligning with BDD principles to promote collaboration among stakeholders, testers, and developers.

Sample of the Step Definition File

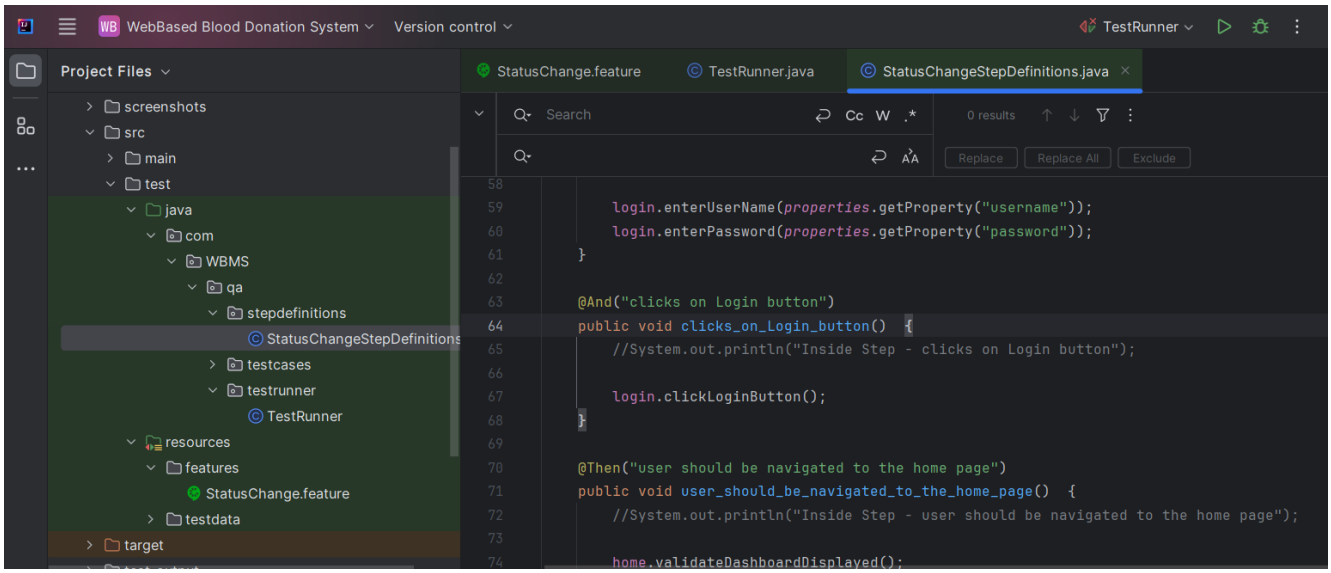


Figure 5.2 Sample Step Definition File

Figure 5.2 presents a Sample Step Definition File, illustrating how each feature outlined in the BDD Feature File is translated into executable steps. These step definitions bridge the high-level scenarios described in the feature file and the automation code that interacts with the application through Selenium WebDriver.

Sample of the TestRunner File

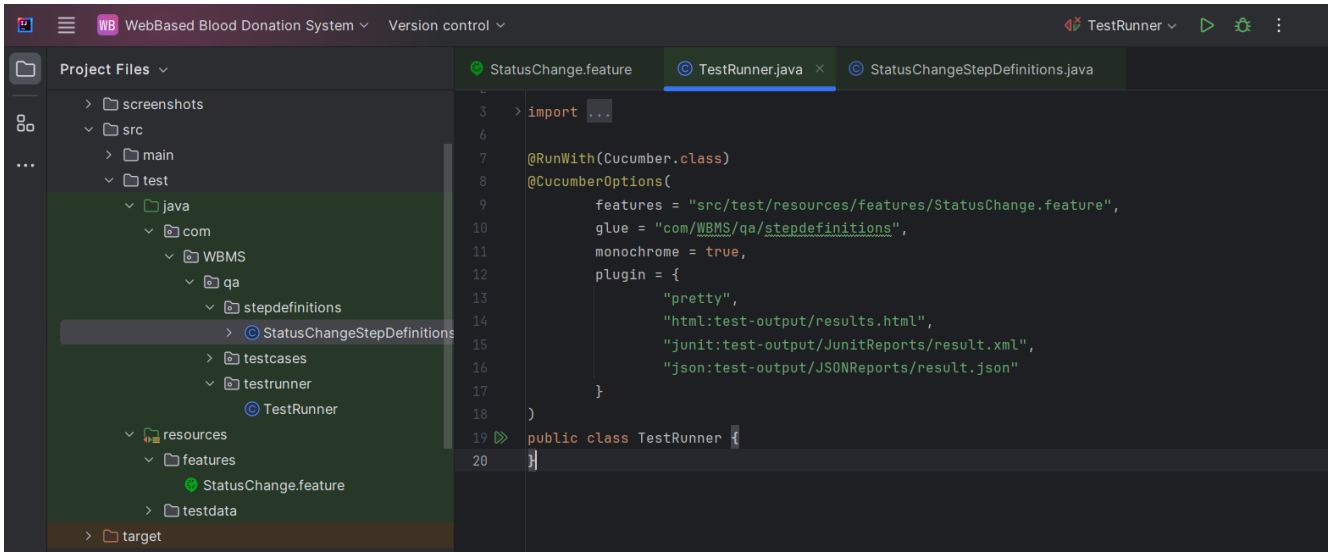


Figure 5.3 Sample Test Runner File

The TestRunner File, depicted in Figure 5.3, acts as the orchestrator that executes the automated tests defined in the step definition file. It coordinates the execution flow, manages test environments, and generates comprehensive test reports, ensuring seamless integration and monitoring of the testing process.

Sample Test Results in HTML Formal

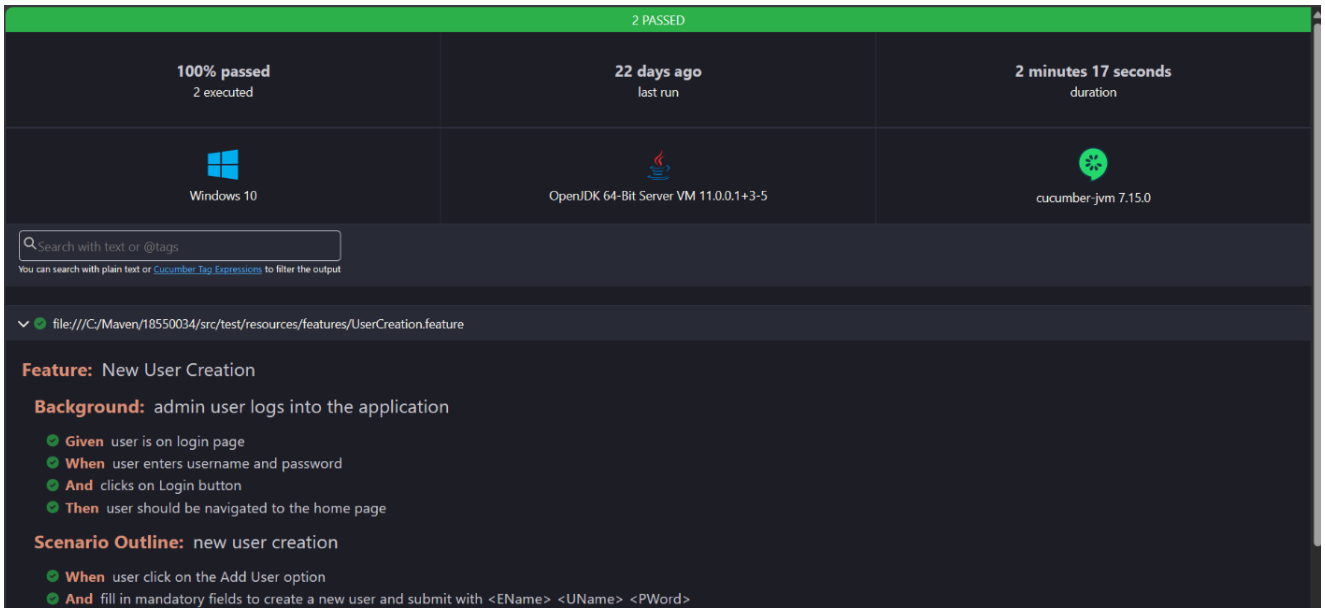


Figure 5.4 Sample Test Runner File

Figure 5.4 showcases a sample Test Runner File with test results in HTML format. This structured and visually informative report provides detailed insights into test outcomes, including passed and failed scenarios, execution times, and any errors or exceptions encountered. Such reports are instrumental in gauging the effectiveness and reliability of automated Selenium testing for blood depot status changes.

By leveraging automated testing methodologies, specifically with Selenium WebDriver and BDD frameworks, organizations can streamline the validation of blood depot status changes, enhance test coverage, and improve overall testing efficiency and accuracy. Combining human-readable test scenarios, executable step definitions, efficient test execution management, and informative test reports contributes significantly to robust and reliable testing practices in blood depot management systems.

5.6 Summary

This chapter evaluates the project's testing processes, covering manual and automated testing methodologies using BDD Cucumber with Selenium WebDriver. It outlines the testing procedures, including the Usability Report, specific test cases, and their results. Chapter 5 provides an overview of the testing evaluation process.

Chapter 6 : CONCLUSION

6.1 Introduction

This chapter concludes the dissertation with an evaluation of the system and whether the project objectives are met or not. It contains details about the lessons learned from the project and future enhancements to the system.

6.2 Work Carried Out

The development of the Web-Based Blood Management System involved several critical tasks and activities to ensure its successful implementation. These tasks included:

- Requirement gathering: Extensive research was conducted to understand the needs of blood donation and distribution systems, including studying existing manual processes and identifying areas for improvement.
- System design: The system architecture, database structure, and user interface were designed to be intuitive, user-friendly, and accessible via web browsers on Windows-based devices.
- Feature implementation: The system's core features, such as donor registration, blood inventory management, and recipient matching, as well as reward points utilizations, were implemented according to the identified requirements.
- Testing and validation: Rigorous testing and validation processes were carried out to ensure the system's functionality, security, and performance met the desired standards.
- Documentation: Comprehensive documentation was created to guide users, administrators, and developers in using, managing, and maintaining the system effectively.

6.3 Limitations

Despite the successful development and implementation of the Web-Based Blood Management System, certain limitations were encountered during the process, including:

- Scope constraints: The system's initial development phase did not include direct integrations with other hospitals and blood collection centres, which could limit its reach and functionality in the early stages.
- Technical dependencies: The system's real-time access to the database may experience delays or dependencies due to network issues or processing times, affecting the responsiveness of certain features.

6.4 Assumptions Made

Throughout the development of the Web-Based Blood Management System, several assumptions were made to guide the design and implementation process, including:

- Each user/entity will have a unique email address for identification within the system.
- System administrators can manage and remove fraudulent or bogus donors from the system.
- Recipients and donors will be prioritized based on a first-in, first-out basis to ensure fairness in blood distribution.

6.5 Lessons Learned

The development of the Web-Based Blood Management System provided valuable insights and lessons learned, including:

Technology selection: Initially considering the WAMP (Windows, Apache, MySQL, PHP) stack for backend development, the project highlighted the importance of selecting appropriate technologies based on industry standards and performance considerations. The decision was

made to stick with PHP and MySQL, ensuring compatibility and reliability in the system's infrastructure.

User interface design: The intuitive and user-friendly design was prioritized to accommodate users with minimal technical knowledge, enhancing overall usability and adoption rates.

System scalability: Future enhancements and scalability considerations, such as mobile application integration and geo-location features were identified to improve system capabilities and meet evolving user needs.

These lessons learned will inform future development efforts and enhancements to ensure the Web-Based Blood Management System remains effective, efficient, and adaptable to changing healthcare requirements.

6.6 Summary

The Conclusion chapter summarizes the tasks carried out, limitations encountered, assumptions made, and lessons learned during the development process. The text concludes by presenting recommendations for future enhancements and expansions of the system further to improve its functionality and utility in the healthcare domain.

REFERENCES

Donate Blood [WWW Document], n.d. URL http://www.nbts.health.gov.lk/index.php/donate-blood (accessed 3.7.24).

fake sendmail for windows [WWW Document], n.d. URL https://www.glob.com.au/sendmail/ (accessed 3.7.24).

Haemonetics® Blood Center Solutions | Manage Blood Collection Safely [WWW Document], n.d. URL https://bloodcenter.haemonetics.com/ (accessed 3.7.24).

Haemonetics Announces Sale Of U.S. Blood Donor Management Software Solution Assets To The GPI Group | Haemonetics Corporation [WWW Document], n.d. URL https://haemonetics.gcs-web.com/news-releases/news-release-details/haemonetics-announces-sale-us-blood-donor-management-software/ (accessed 3.7.24).

Haemonetics Presentation | PPT [WWW Document], n.d. URL https://www.slideshare.net/rleiner/haemoneticspresentation-1556587 (accessed 3.7.24).

Hashim, S.A., Al-Madani, A.M., Al-Amri, S.M., Al-Ghamdi, A.M., Bashamakh, B.S., Aljojo, N., 2014. Online blood donation reservation and managementsystem in Jeddah 11, 60–65.

Mussema, A., Bawore, S.G., Seid, A.M., 2023. Voluntary blood donation knowledge, attitude, and practice among adult populations of Hosanna Town, South Ethiopia: a community-based cross-sectional study. Front. Public Health 11. https://doi.org/10.3389/fpubh.2023.1141544

What is a 3-Tier Application Architecture? Definition from SearchSoftwareQuality [WWW Document], n.d. . Software Quality. URL https://www.techtarget.com/searchsoftwarequality/definition/3-tier-application (accessed 3.7.24).

APPENDIX A

System Documentation

❖ WAMP server Installation (Installation Steps are available in figure A.1 – A.4)

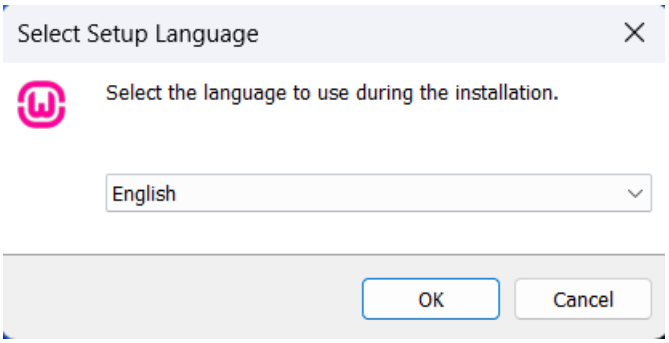


Figure A.1 WAMP Installation first step window

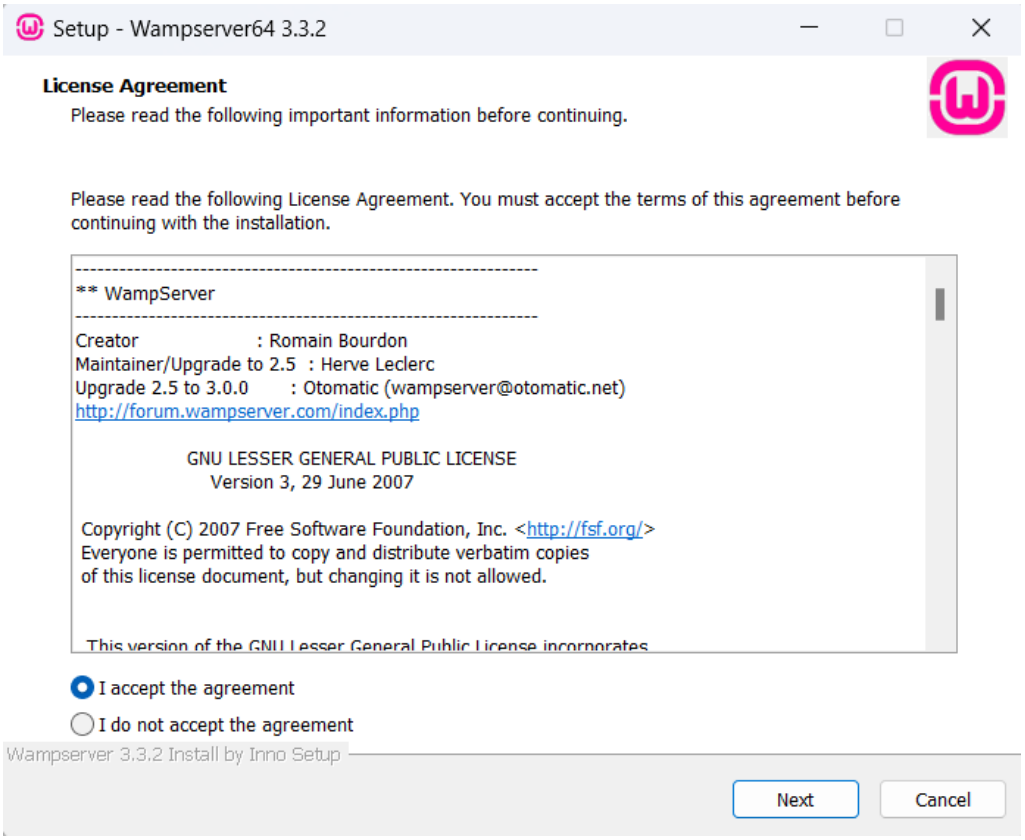


Figure A.2 WAMP Installation second step window

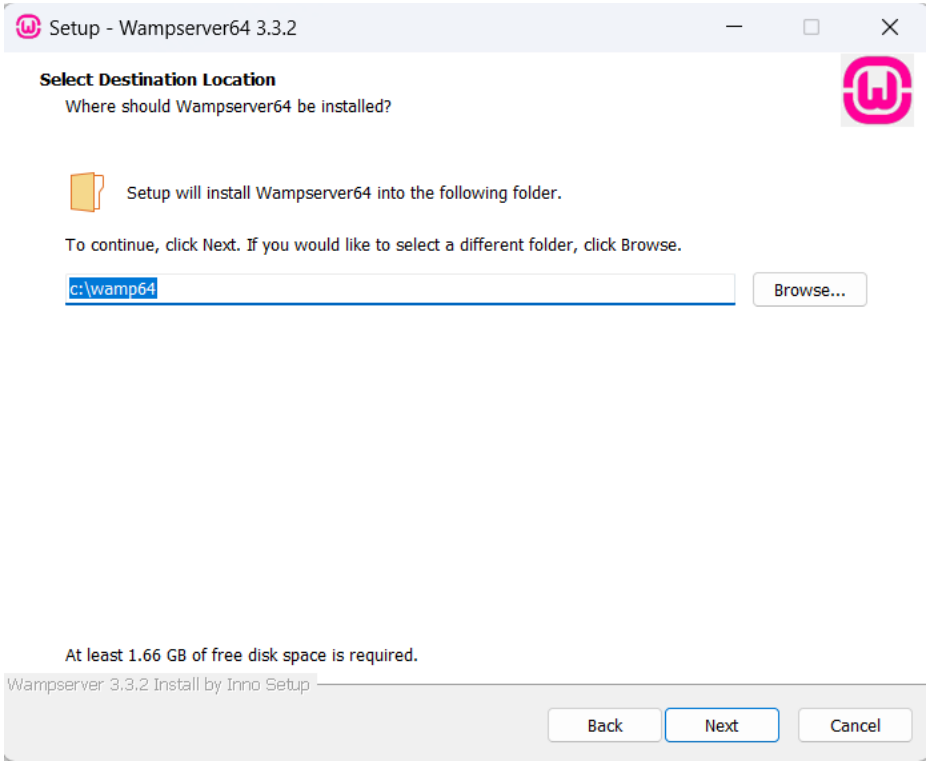


Figure A.3 WAMP Installation third step window

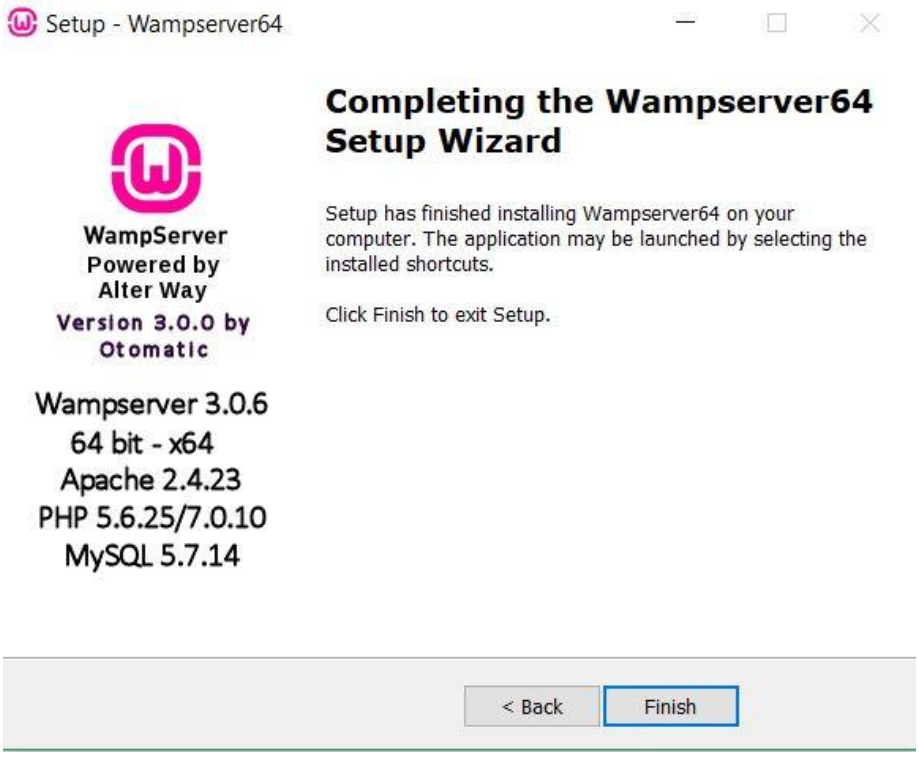


Figure A.4 WAMP Installation fourth step window

❖ Change the Root Password of the MYSQL database (refer figure A.4)

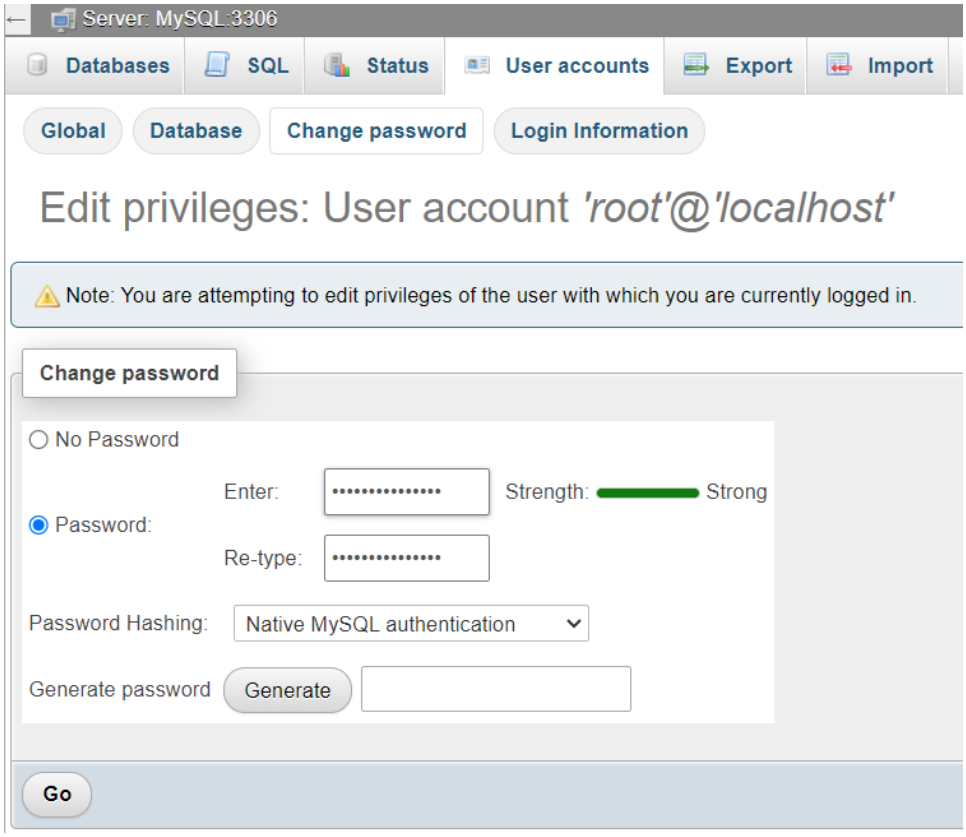


Figure A.5 Change root password of database

❖ Deploy the Application sources to www folder in the wamp64 directory (refer figure A.5)

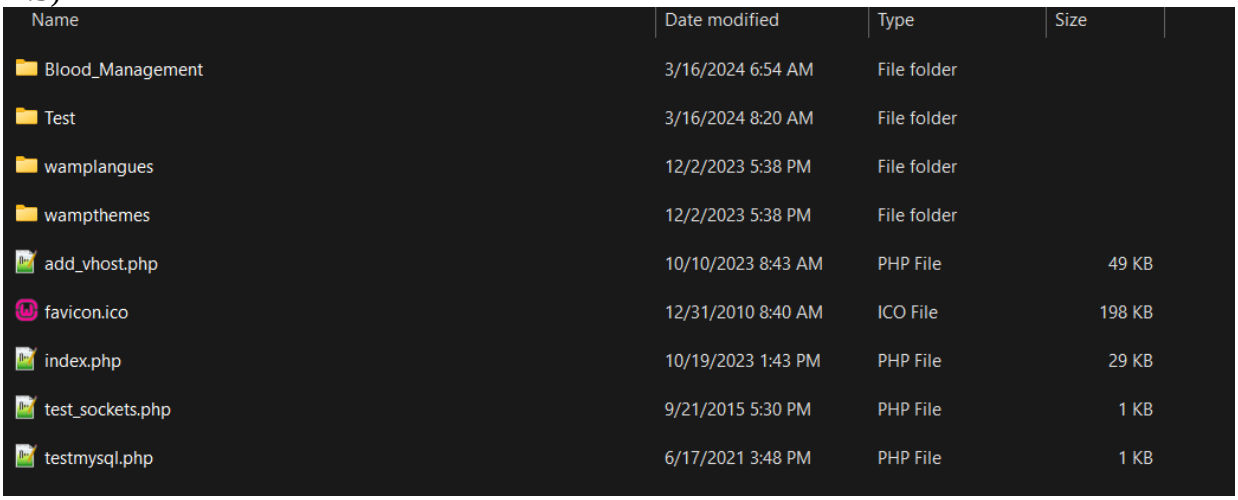


Figure A.6 Deploy application sources

❖ Deploy the sendmail folder to wamp64 directory (http://glob.com.au/sendmail/) (refer figure A.6)

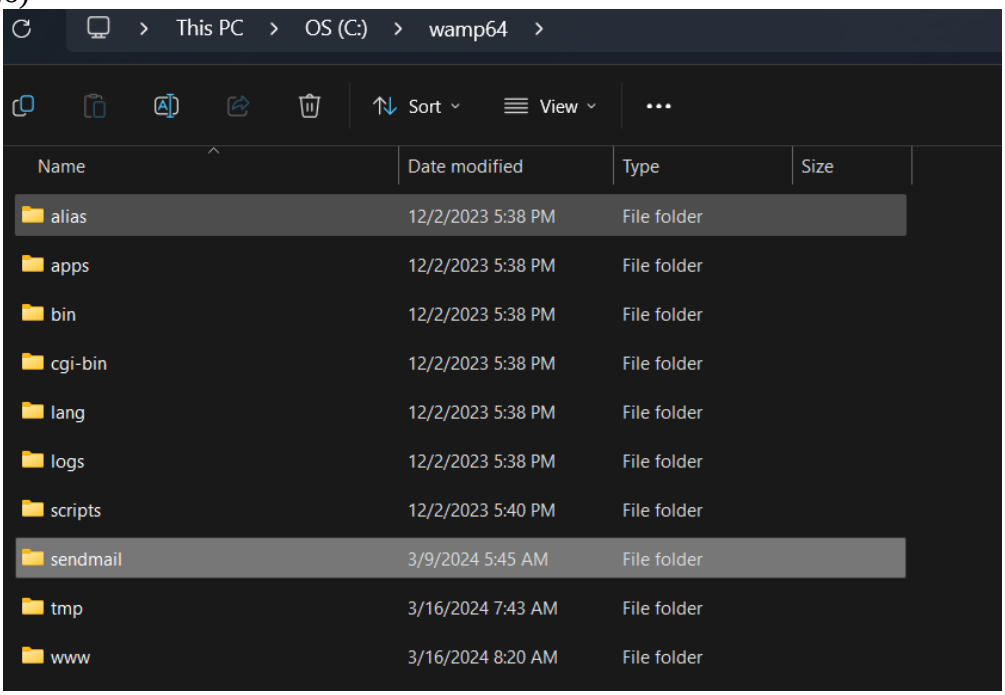


Figure A.7 Sendmail folder deployment

❖ Change the sendmail.ini file as below (refer figure A.7)

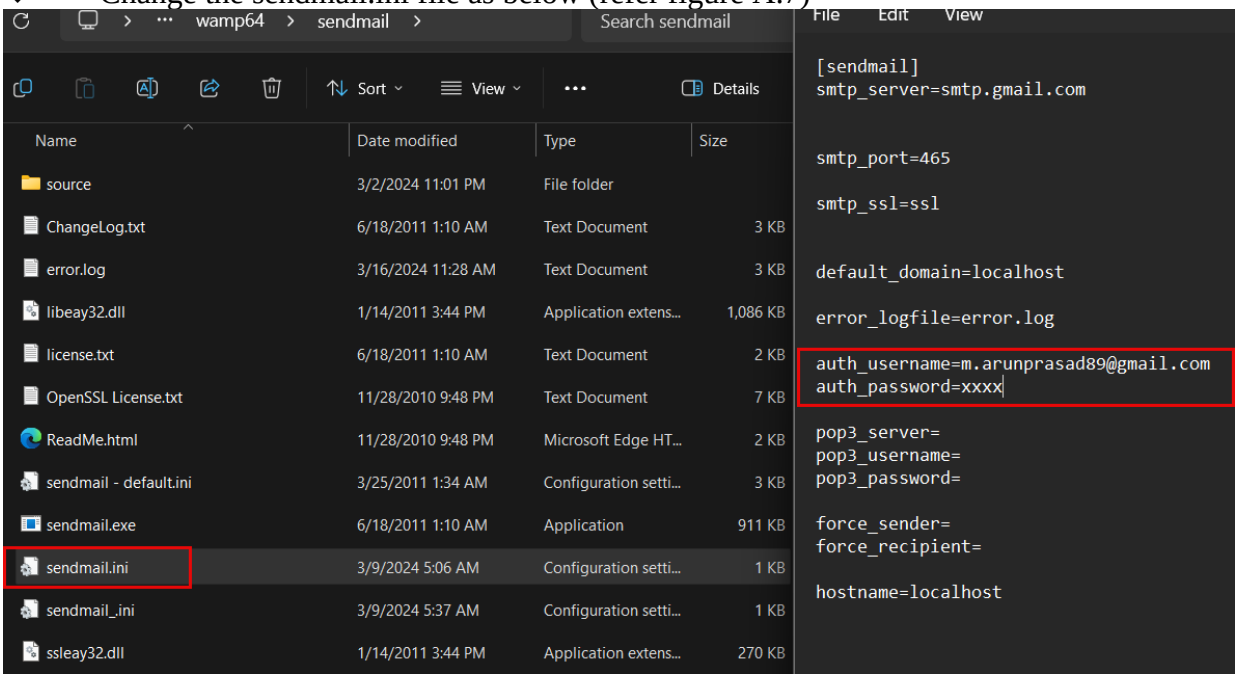


Figure A.8 Sendmail.ini modification

❖ Do the following changes in apache server php.ini file

➢ sendmail_path ="D:\wamp64\sendmail\sendmail.exe -t -i")

❖ Install MySQL Workbench

(refer installation steps in figure A.9, A.10)

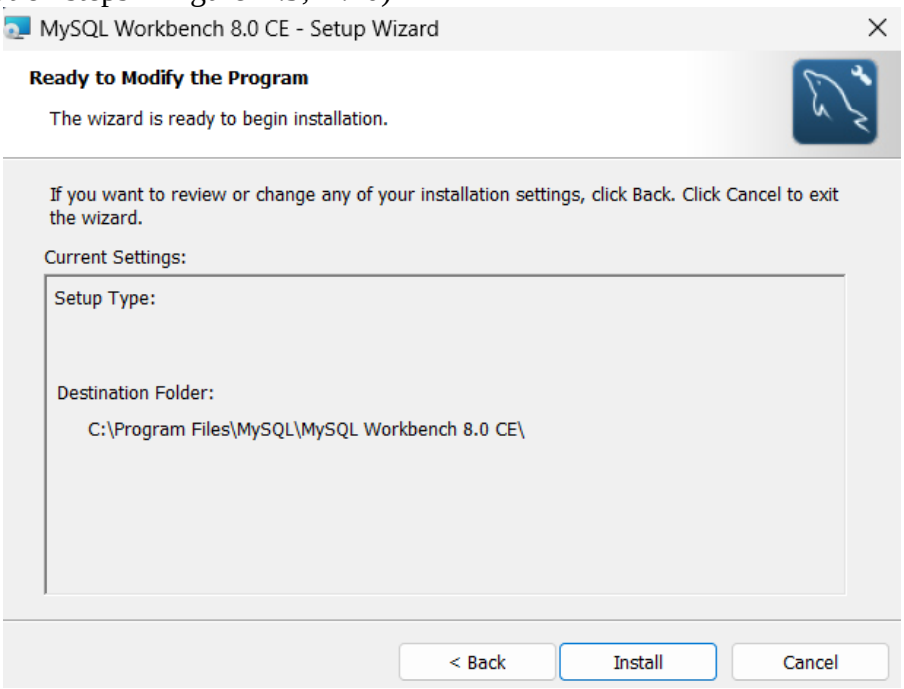


Figure A.9 MySQL workbench installation first step window

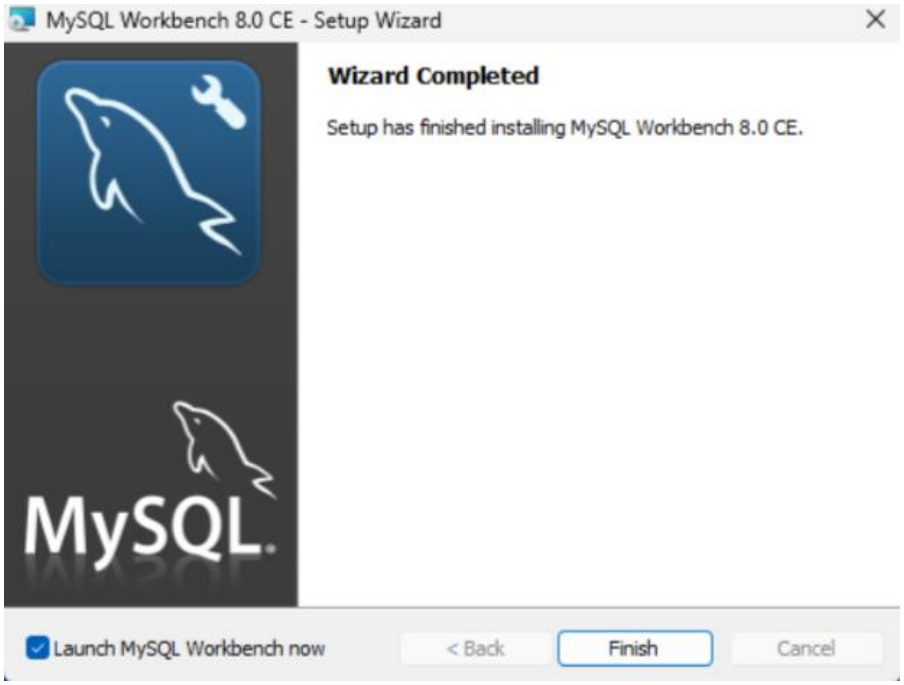


Figure A.10 MySQL workbench installation second step window

Import the backup of the database via data import option in MySQL workbench

(refer steps in figure A.11, A.12)

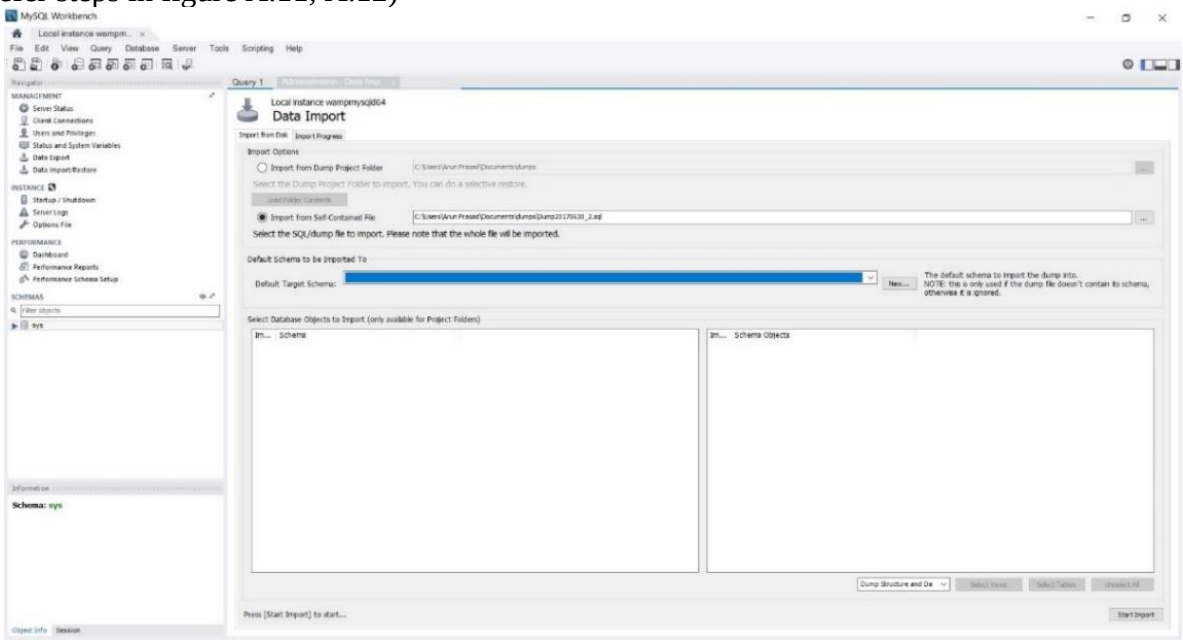


Figure A.11 Database Import First step window

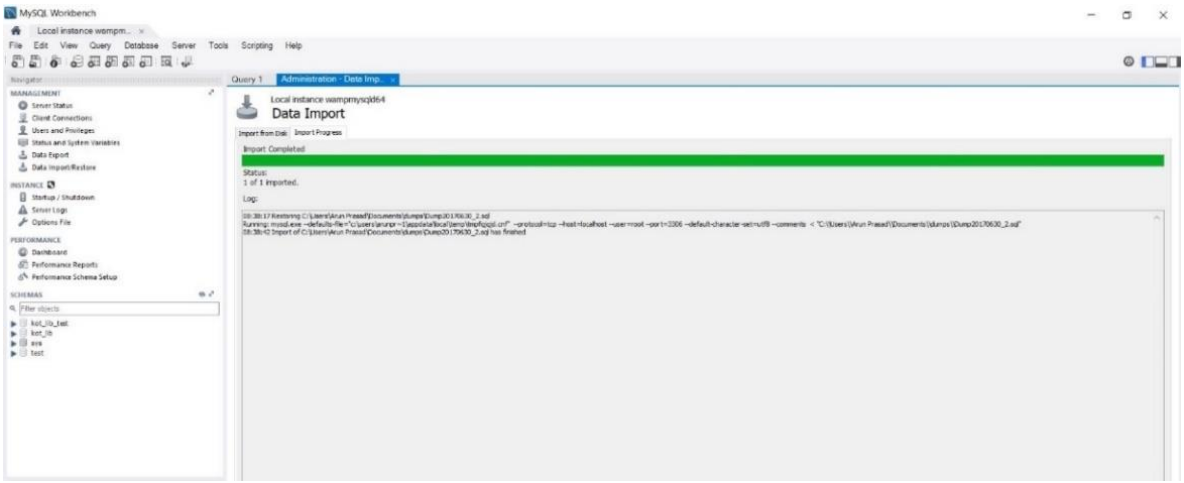


Figure A.12 Database Import second step window

APPENDIX B

Design Documentation

Database Design and Table Structures

Tables are designed to adhere to the third normal form as much as possible.

Administrator related Master data will be stored in admin_master_tbl (see figure B.1).

Table Name: admin_master_tbl Schema: **blood_management**

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ADMIN_ID	VARCHAR(12)	☑	☑	☐	☐	☐	☐	☐	☐	
TITLE	CHAR(5)	☐	☑	☐	☐	☐	☐	☐	☐	'MR.'
FULL_NAME	VARCHAR(60)	☐	☑	☐	☐	☐	☐	☐	☐	
NIC_NO	VARCHAR(12)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
DEL_FLG	ENUM('Y', 'N')	☐	☑	☐	☐	☐	☐	☐	☐	'N'
ADDRESS	VARCHAR(60)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
EMAIL_ADDRESS	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	
TP_NO	VARCHAR(20)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
MOBILE_NO	VARCHAR(20)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
R_CRE_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
R_CRE_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'
L_CHG_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
L_CHG_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'

Figure B.1 Admin master table structure

Administrator related Login details will be stored in admin_login_tbl (see figure B.2).

Table Name: admin_login_tbl Schema: **blood_management**

Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
ADMIN_ID	VARCHAR(12)	☑	☑	☐	☐	☐	☐	☐	☐	
USER_PWD	VARCHAR(40)	☐	☑	☐	☐	☐	☐	☐	☐	
NO_OF_ATTEMPTS	SMALLINT	☐	☑	☐	☐	☐	☐	☐	☐	'0'
ROLE	ENUM('SUPERADMIN...	☐	☑	☐	☐	☐	☐	☐	☐	'ADMIN'
IMG_NAME	VARCHAR(60)	☐	☑	☐	☐	☐	☐	☐	☐	
SECURITY_QUESTION	VARCHAR(60)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
SECURITY_ANSWER	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
R_CRE_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
R_CRE_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'
L_CHG_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON...
L_CHG_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'

Figure B.2 Admin Login table

User master table (see figure B.3) contains the main User details. Whenever User record is created, just before the table insertion, User ID will be auto generated through database trigger

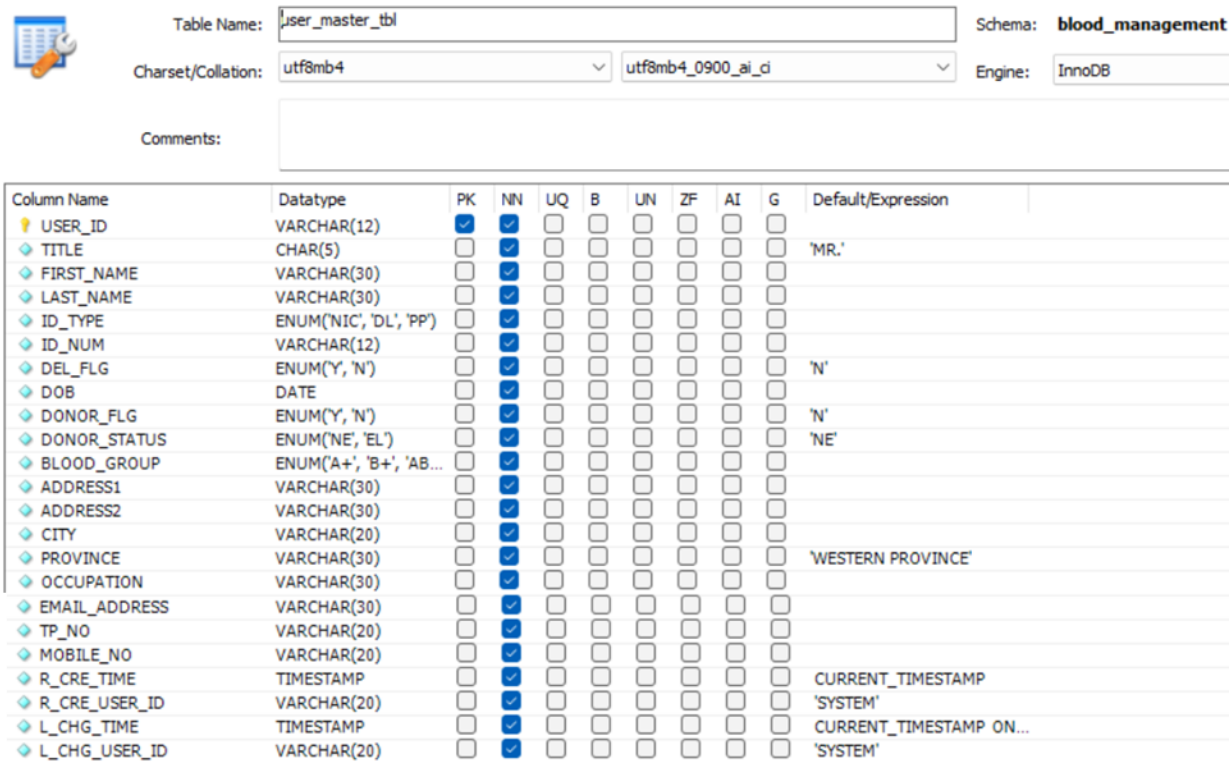


Table Name: user_master_tbl
Schema: blood_management
Charset/Collation: utf8mb4 / utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
USER_ID	VARCHAR(12)	☑	☑	☐	☐	☐	☐	☐	☐	
TITLE	CHAR(5)	☐	☑	☐	☐	☐	☐	☐	☐	'MR.'
FIRST_NAME	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	
LAST_NAME	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	
ID_TYPE	ENUM('NIC', 'DL', 'PP')	☐	☑	☐	☐	☐	☐	☐	☐	
ID_NUM	VARCHAR(12)	☐	☑	☐	☐	☐	☐	☐	☐	
DEL_FLG	ENUM('Y', 'N')	☐	☑	☐	☐	☐	☐	☐	☐	'N'
DOB	DATE	☐	☑	☐	☐	☐	☐	☐	☐	
DONOR_FLG	ENUM('Y', 'N')	☐	☑	☐	☐	☐	☐	☐	☐	'N'
DONOR_STATUS	ENUM('NE', 'EL')	☐	☑	☐	☐	☐	☐	☐	☐	'NE'
BLOOD_GROUP	ENUM('A+', 'B+', 'AB...	☐	☑	☐	☐	☐	☐	☐	☐	
ADDRESS1	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	
ADDRESS2	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	
CITY	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	
PROVINCE	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	'WESTERN PROVINCE'
OCCUPATION	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	
EMAIL_ADDRESS	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	
TP_NO	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	
MOBILE_NO	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	
R_CRE_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
R_CRE_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'
L_CHG_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON...
L_CHG_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'

Figure B.3 User master table

All User related information is available in user_detail_tbl (see figure B.4)

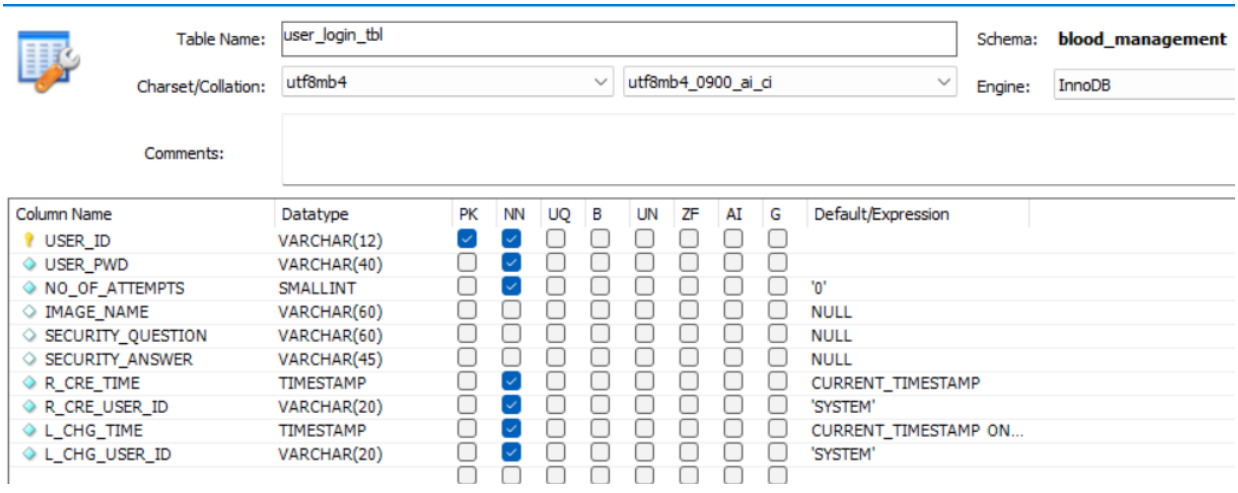


Table Name: user_login_tbl
Schema: blood_management
Charset/Collation: utf8mb4 / utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
USER_ID	VARCHAR(12)	☑	☑	☐	☐	☐	☐	☐	☐	
USER_PWD	VARCHAR(40)	☐	☑	☐	☐	☐	☐	☐	☐	
NO_OF_ATTEMPTS	SMALLINT	☐	☑	☐	☐	☐	☐	☐	☐	'0'
IMAGE_NAME	VARCHAR(60)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
SECURITY_QUESTION	VARCHAR(60)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
SECURITY_ANSWER	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
R_CRE_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
R_CRE_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'
L_CHG_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON...
L_CHG_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'

Figure B.4 User Login table

All Blood Depot related information is available in blood_depot_master_tbl (see figure B.5)

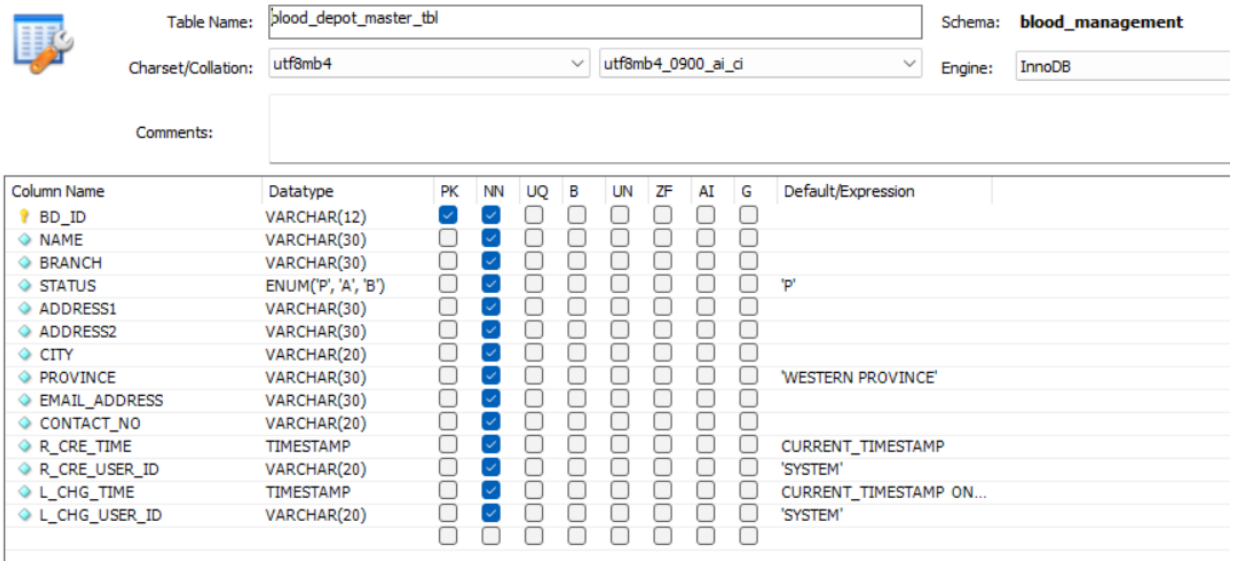


Table Name: blood_depot_master_tbl
Schema: blood_management
Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
BD_ID	VARCHAR(12)	☑	☑	☐	☐	☐	☐	☐	☐	
NAME	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	
BRANCH	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	
STATUS	ENUM('P', 'A', 'B')	☐	☑	☐	☐	☐	☐	☐	☐	'P'
ADDRESS1	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	
ADDRESS2	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	
CITY	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	
PROVINCE	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	'WESTERN PROVINCE'
EMAIL_ADDRESS	VARCHAR(30)	☐	☑	☐	☐	☐	☐	☐	☐	
CONTACT_NO	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	
R_CRE_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
R_CRE_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'
L_CHG_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON...
L_CHG_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'

Figure B.5 Blood_depot_master table

Blood Depot related Login details will be stored in blood_depot_login_tbl (see figure B.6).

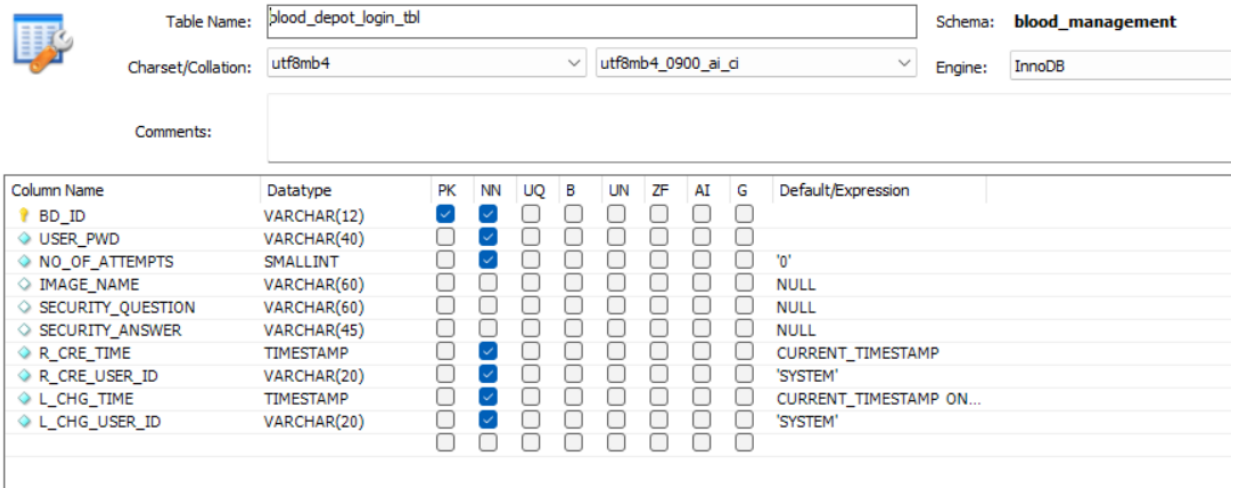


Table Name: blood_depot_login_tbl
Schema: blood_management
Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci
Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
BD_ID	VARCHAR(12)	☑	☑	☐	☐	☐	☐	☐	☐	
USER_PWD	VARCHAR(40)	☐	☑	☐	☐	☐	☐	☐	☐	
NO_OF_ATTEMPTS	SMALLINT	☐	☑	☐	☐	☐	☐	☐	☐	'0'
IMAGE_NAME	VARCHAR(60)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
SECURITY_QUESTION	VARCHAR(60)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
SECURITY_ANSWER	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
R_CRE_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
R_CRE_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'
L_CHG_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON...
L_CHG_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'

Figure B.6 Blood_depot_login table

Donor related Registration related details will be stored in donor_tbl (see figure B.7).

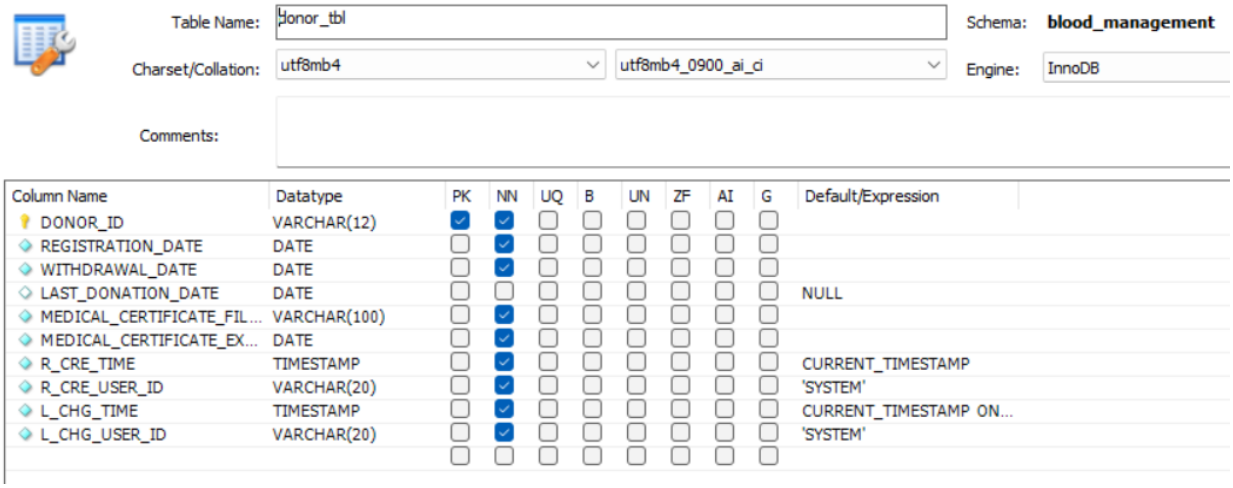


Figure B.7 Donor table

When Blood Depot Host a Campaign, those Campaign related details are maintained in this table

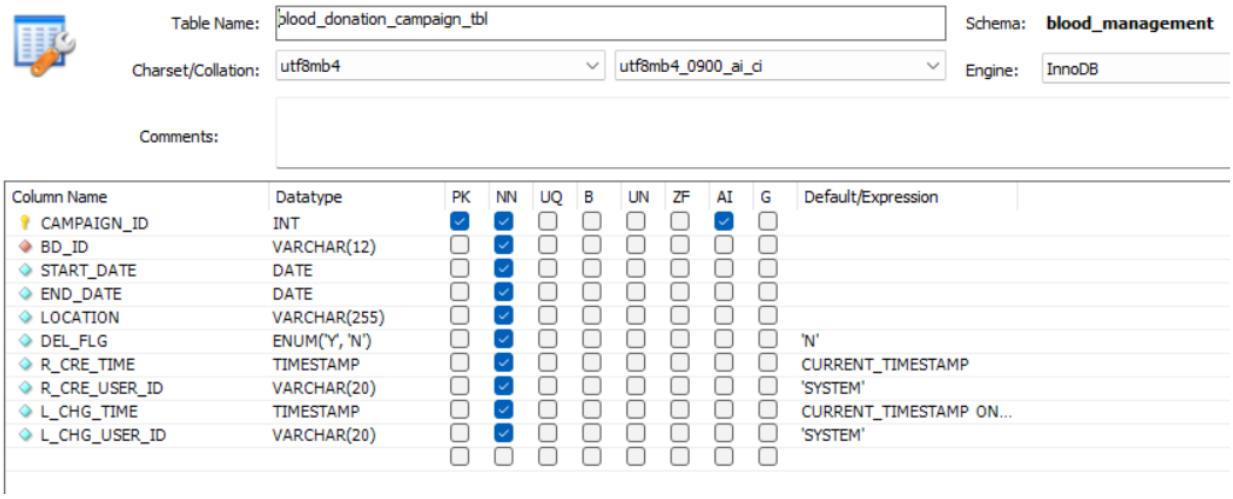


Figure B.8 Donation Campaign table

Each Blood Depot wise Blood Stock details are maintained in this table

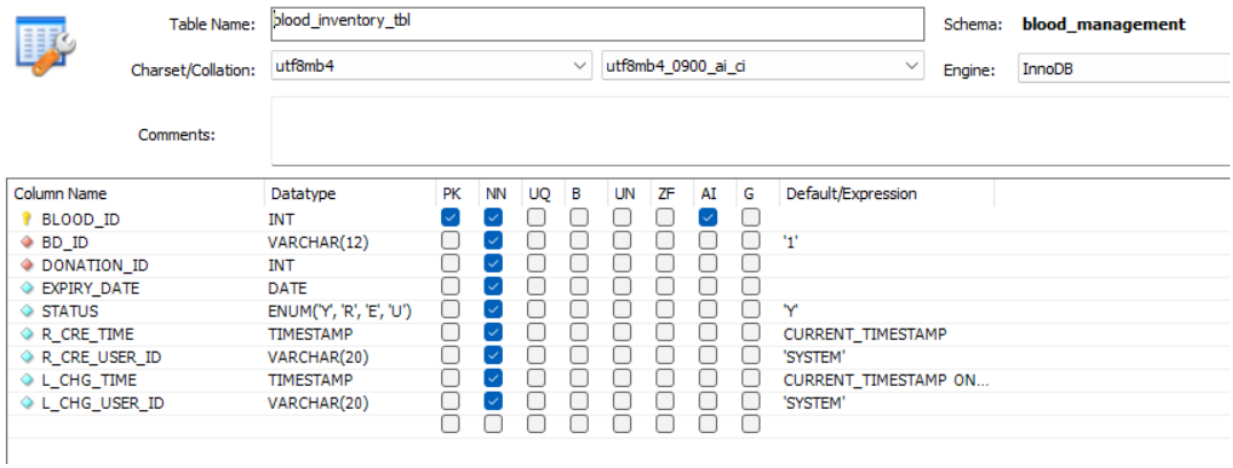


Table Name: blood_inventory_tbl Schema: blood_management
Charset/Collation: utf8mb4 / utf8mb4_0900_ai_ci Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
BLOOD_ID	INT	☑	☑	☐	☐	☐	☐	☑	☐	
BD_ID	VARCHAR(12)	☐	☑	☐	☐	☐	☐	☐	☐	'1'
DONATION_ID	INT	☐	☑	☐	☐	☐	☐	☐	☐	
EXPIRY_DATE	DATE	☐	☑	☐	☐	☐	☐	☐	☐	
STATUS	ENUM('Y', 'R', 'E', 'U')	☐	☑	☐	☐	☐	☐	☐	☐	'Y'
R_CRE_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
R_CRE_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'
L_CHG_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON...
L_CHG_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'

Figure B.9 Blood Inventory table

When a Recipient requests for Blood Request, those details are maintained in this table

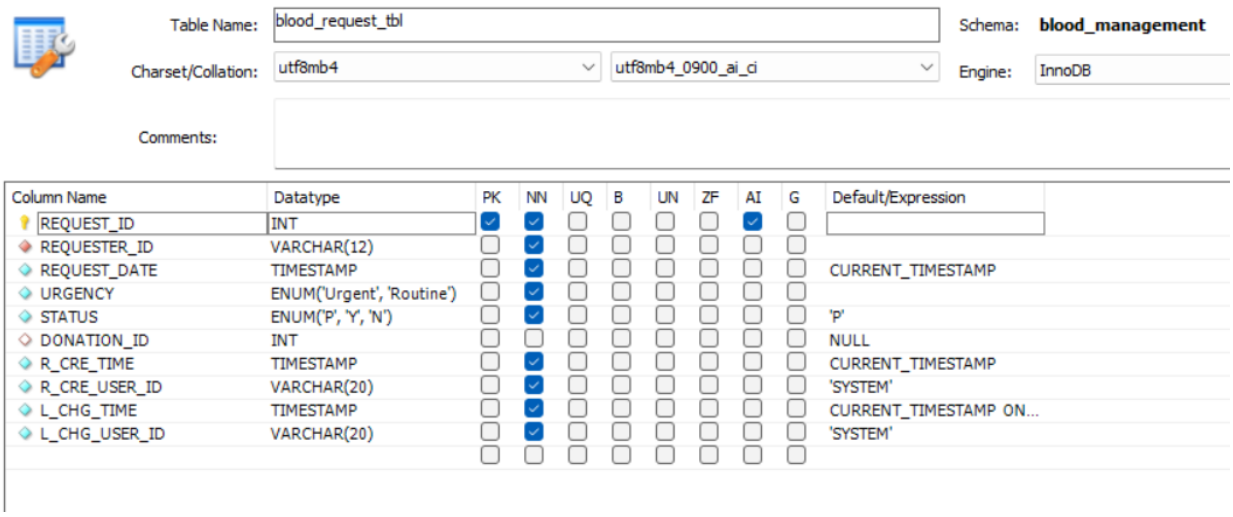
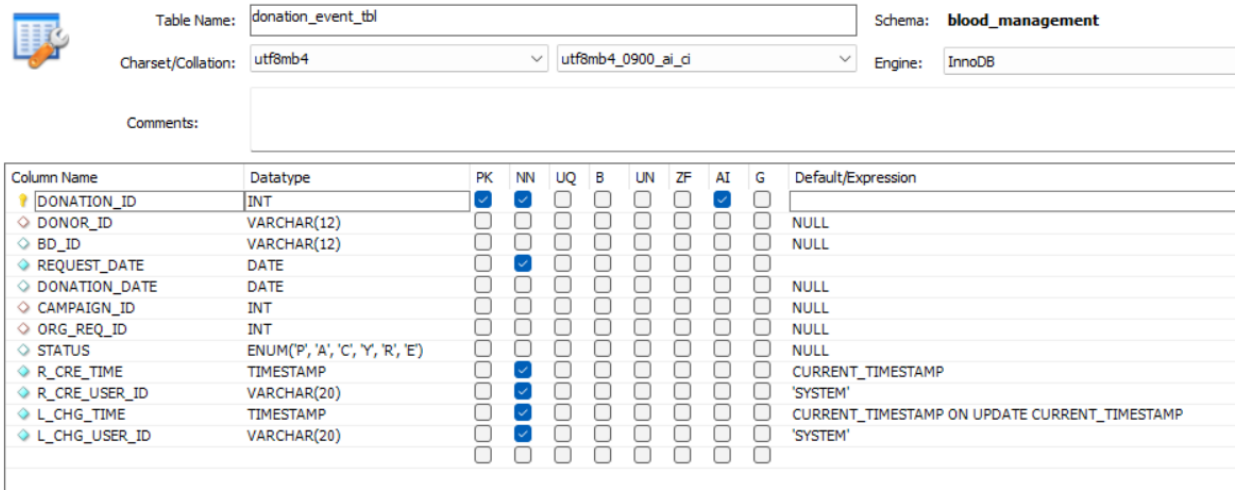


Table Name: blood_request_tbl Schema: blood_management
Charset/Collation: utf8mb4 / utf8mb4_0900_ai_ci Engine: InnoDB
Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
REQUEST_ID	INT	☑	☑	☐	☐	☐	☐	☑	☐	
REQUESTER_ID	VARCHAR(12)	☐	☑	☐	☐	☐	☐	☐	☐	
REQUEST_DATE	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
URGENCY	ENUM('Urgent', 'Routine')	☐	☑	☐	☐	☐	☐	☐	☐	
STATUS	ENUM('P', 'Y', 'N')	☐	☑	☐	☐	☐	☐	☐	☐	'P'
DONATION_ID	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
R_CRE_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
R_CRE_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'
L_CHG_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON...
L_CHG_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'

Figure B.10 Blood Request table

Blood Donation Event related details are maintained in this table

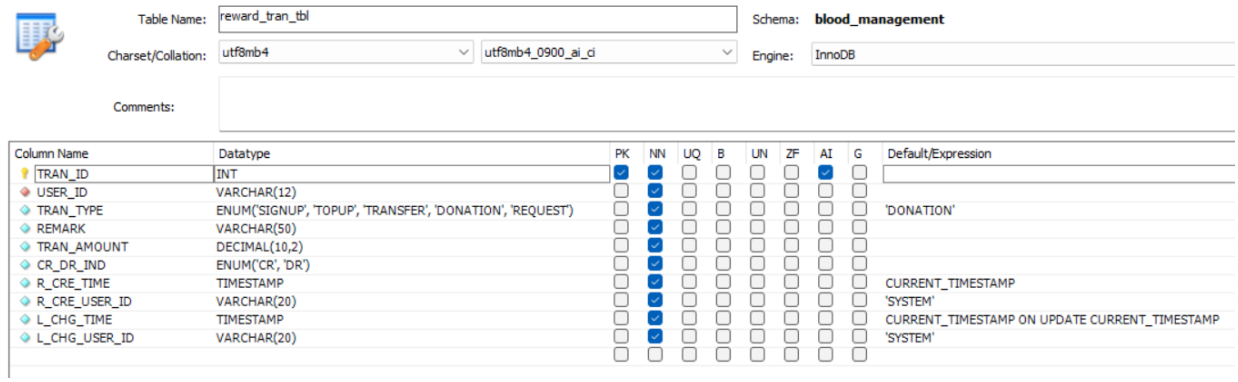


Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
DONATION_ID	INT	✓	✓					✓		
DONOR_ID	VARCHAR(12)									NULL
BD_ID	VARCHAR(12)									NULL
REQUEST_DATE	DATE		✓							
DONATION_DATE	DATE									NULL
CAMPAIGN_ID	INT									NULL
ORG_REQ_ID	INT									NULL
STATUS	ENUM('P', 'A', 'C', 'Y', 'R', 'E')									NULL
R_CRE_TIME	TIMESTAMP		✓							CURRENT_TIMESTAMP
R_CRE_USER_ID	VARCHAR(20)		✓							'SYSTEM'
L_CHG_TIME	TIMESTAMP		✓							CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
L_CHG_USER_ID	VARCHAR(20)		✓							'SYSTEM'

Figure B.11 Blood Donation Event table

Other Important table structures

Figures B.12- B.13 illustrates table structures of other important User reward related tables such as reward_tran_tbl, reward_balance_table)



Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
TRAN_ID	INT	✓	✓					✓		
USER_ID	VARCHAR(12)		✓							
TRAN_TYPE	ENUM('SIGNUP', 'TOPUP', 'TRANSFER', 'DONATION', 'REQUEST')		✓							'DONATION'
REMARK	VARCHAR(50)		✓							
TRAN_AMOUNT	DECIMAL(10,2)		✓							
CR_DR_IND	ENUM('CR', 'DR')		✓							
R_CRE_TIME	TIMESTAMP		✓							CURRENT_TIMESTAMP
R_CRE_USER_ID	VARCHAR(20)		✓							'SYSTEM'
L_CHG_TIME	TIMESTAMP		✓							CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
L_CHG_USER_ID	VARCHAR(20)		✓							'SYSTEM'

Figure B.12 Reward Transaction table

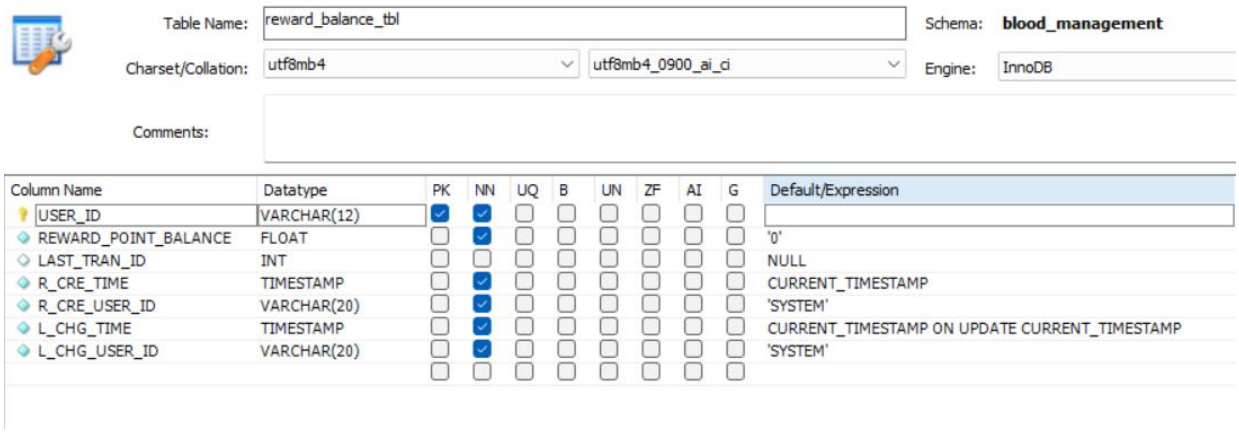


Table Name: reward_balance_tbl Schema: blood_management

Charset/Collation: utf8mb4 utf8mb4_0900_ai_ci Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
USER_ID	VARCHAR(12)	☑	☑	☐	☐	☐	☐	☐	☐	
REWARD_POINT_BALANCE	FLOAT	☐	☑	☐	☐	☐	☐	☐	☐	'0'
LAST_TRAN_ID	INT	☐	☐	☐	☐	☐	☐	☐	☐	NULL
R_CRE_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
R_CRE_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'
L_CHG_TIME	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
L_CHG_USER_ID	VARCHAR(20)	☐	☑	☐	☐	☐	☐	☐	☐	'SYSTEM'
		☐	☐	☐	☐	☐	☐	☐	☐	

Figure B.13 Reward Balance table

APPENDIX C

Code Listing

Following code (see figure C.1) is used to upload User picture to the system and updates the table with the name of the image file.

```
//uploaded Pic details
    $pic = mysqli_real_escape_string($dbc, trim($_FILES['PIC']['name']));
    $pic_size = $_FILES['PIC']['size'];
    $pic_type = $_FILES['PIC']['type'];


    if (($pic_type == 'image/gif') || ($pic_type == 'image/jpeg') || ($pic_type == 'image/png') || ($pic_type == 'image/pjpeg')){
       if (($pic_size > 0) && ($pic_size <= $max_size)) {
       if ($_FILES['PIC']['error'] == 0) {

          $pic_name=time(). $pic;
         // Move the file to the target upload folder
         $img_location = $mem_im_loc.$pic_name;
         if (move_uploaded_file($_FILES['PIC']['tmp_name'], $img_location)) {
           // Write the data to the database
           $query2 = "UPDATE user_login_tbl SET image_name='$pic_name' where USER_ID='$user_id'";
                     //echo "<DIV CLASS='debug'>$query2</DIV>";

            $result2=mysqli_query($dbc, $query2);

             if ($result2){
                 @unlink($mem_im_loc.$org_pic);
                    $org_pic=$pic_name;
             }
         }
         else {
           $err_msg="Problem Encountered while uploading the Image";
         }
       }
    }
        else {
         $err_msg='File size should be less than'.($max_size / 1024).' KB in size';
            }
    }
    else {
       $err_msg="Only GIF, JPEG, or PNG formats are supported";
        }

    // Try to delete the temporary screen shot image file
    @unlink($_FILES['PIC']['tmp_name']);

    mysqli_close($dbc);
}
```

Figure C.1 User picture upload

Following code (see figure C.2) is used to generate email to the Registered Email Address once new User Signup to the System

```php
//uploaded Pic details
    $pic = mysqli_real_escape_string($dbc, trim($_FILES['PIC']['name']));
    $pic_size = $_FILES['PIC']['size'];
    $pic_type = $_FILES['PIC']['type'];


    if (($pic_type == 'image/gif') || ($pic_type == 'image/jpeg') || ($pic_type == 'image/png') || ($pic_type == 'image/pjpeg')){
      if (($pic_size > 0) && ($pic_size <= $max_size)) {
      if ($_FILES['PIC']['error'] == 0) {

          $pic_name=time(). $pic;
        // Move the file to the target upload folder
        $img_location = $mem_im_loc.$pic_name;
        if (move_uploaded_file($_FILES['PIC']['tmp_name'], $img_location)) {
          // Write the data to the database
          $query2 = "UPDATE user_login_tbl SET image_name='$pic_name' where USER_ID='$user_id'";
                          //echo "<DIV CLASS='debug'>$query2</DIV>";

           $result2=mysqli_query($dbc, $query2);

            if ($result2){
                @unlink($mem_im_loc.$org_pic);
                   $org_pic=$pic_name;
            }
        }
        else {
          $err_msg="Problem Encountered while uploading the Image";
        }
      }
    }
        else {
         $err_msg='File size should be less than'.($max_size / 1024).' KB in size';
            }
    }
    else {
       $err_msg="Only GIF, JPEG, or PNG formats are supported";
        }

    // Try to delete the temporary screen shot image file
    @unlink($_FILES['PIC']['tmp_name']);

    mysqli_close($dbc);
```

Figure C.2 User password email generation

Following code (see figure C.3) is used for the random password generation at the time of new User activation

```php
//Age Calculation Before Insert
$query_age = "select datediff(curdate(),STR_TO_DATE('$dob','%Y-%m-%d' ))/365 as AGE from dual";
 $result_age=mysqli_query($dbc, $query_age);
    $row=mysqli_fetch_array($result_age);
    $age=$row['AGE'];
    if ($age>=18){

$password = "";
//random generate password
  for ($i = 0; $i < 8; $i++) {
    $password.= chr(rand(97, 122));
  }


//insert statements begins here
    begin($dbc);
// insert to the user_master_tbl
$query = "INSERT INTO user_master_tbl(TITLE, FIRST_NAME, LAST_NAME, ID_TYPE, ID_NUM, DOB,DONOR_FLG,DONOR_STATUS, BLOOD_GROUP,  ADDRESS1, ADDRESS2, CITY, PROVINCE, OCCUPATION,
          EMAIL_ADDRESS, TP_NO, MOBILE_NO, R_CRE_TIME, R_CRE_USER_ID, L_CHG_TIME, L_CHG_USER_ID) values (
          '$title' ,'$first_name' ,'$last_name' ,'$id_type' ,'$id_num' ,STR_TO_DATE('$dob','%Y-%m-%d'),'$donor', '$blood_group' ,'NE' ,'$address1' ,'$address2' ,'$city' ,
          '$province' , '$occupation' ,
          '$email_address', '$tp_no', '$mobile_no', now(), 'SYSTEM', now(), 'SYSTEM')";
                          echo "<DIV CLASS='debug'>$query</DIV>";

$result=mysqli_query($dbc, $query);
```

Figure C.3 Random password on New User Creation

Below mentioned figure C.4, C.5 portrays the java script validations incorporated in the User registration screen.

```
function validateNIC(inputField, helpText) {

   if ((inputField.value.length != 10) && (inputField.value.length != 12)) {
     if (helpText != null){
       helpText.innerHTML = "NIC should Contain 10 or 12 chars.";}
     return false;
   }

   if (inputField.value.length ==12) {
     if (isNaN(inputField.value)){
       if (helpText != null){
       helpText.innerHTML = "12 Digits NIC should only contain Numbers";}
     return false;
     }
   }

   if (inputField.value.length ==10) {
     if (isNaN(inputField.value.substring(0, 9))){
       if (helpText != null){
       helpText.innerHTML = "First 9 Digits should only contain Numbers";}
     return false;
     }
   }

   if (inputField.value.length ==10) {
     if ((inputField.value.substring(9, 10) != 'V') && (inputField.value.substring(9, 10) != 'X')){
       if (helpText != null){
       helpText.innerHTML = "Last Digit of NIC should be V or X";}
     return false;
          }

       else {
       if (helpText != null)
       helpText.innerHTML = "";
       return true;
       }
   }
   else {
       if (helpText != null)
       helpText.innerHTML = "";
       return true;
```

Figure C.4 JavaScript validations

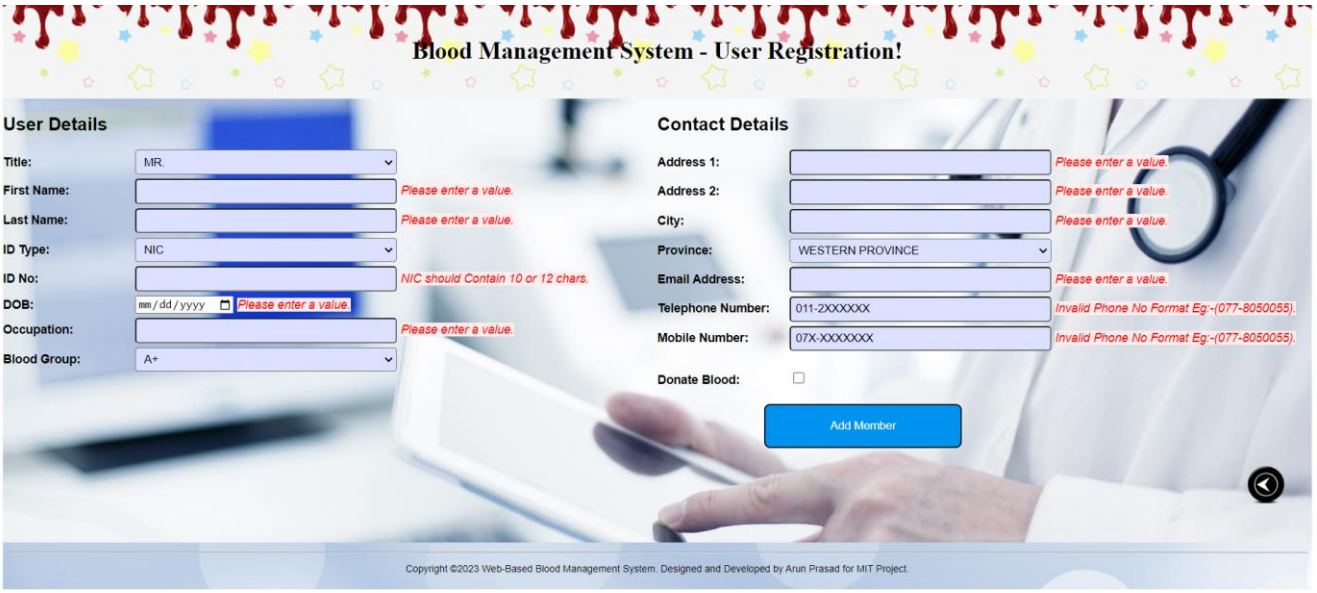


Figure C.5 User registration java script validations

APPENDIX D

Management Reports

All the requested reports are available in the ‘Reports’ section.

Following figure D.1 shows the “View Reports” Option available for Administrator.

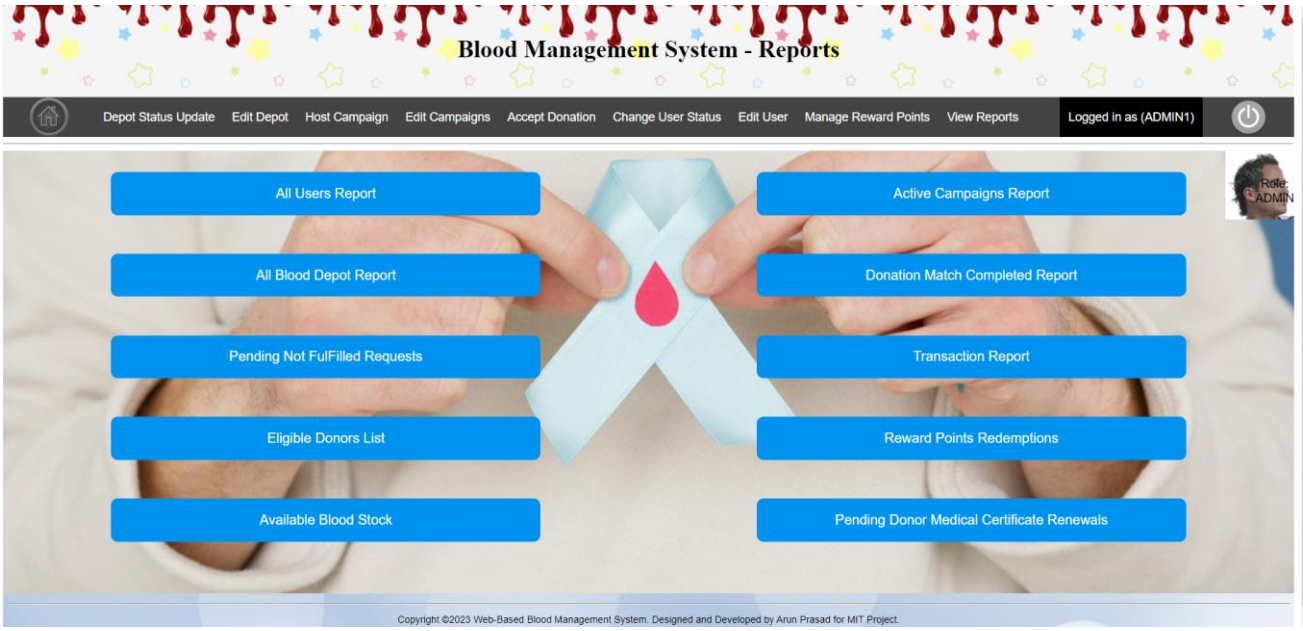
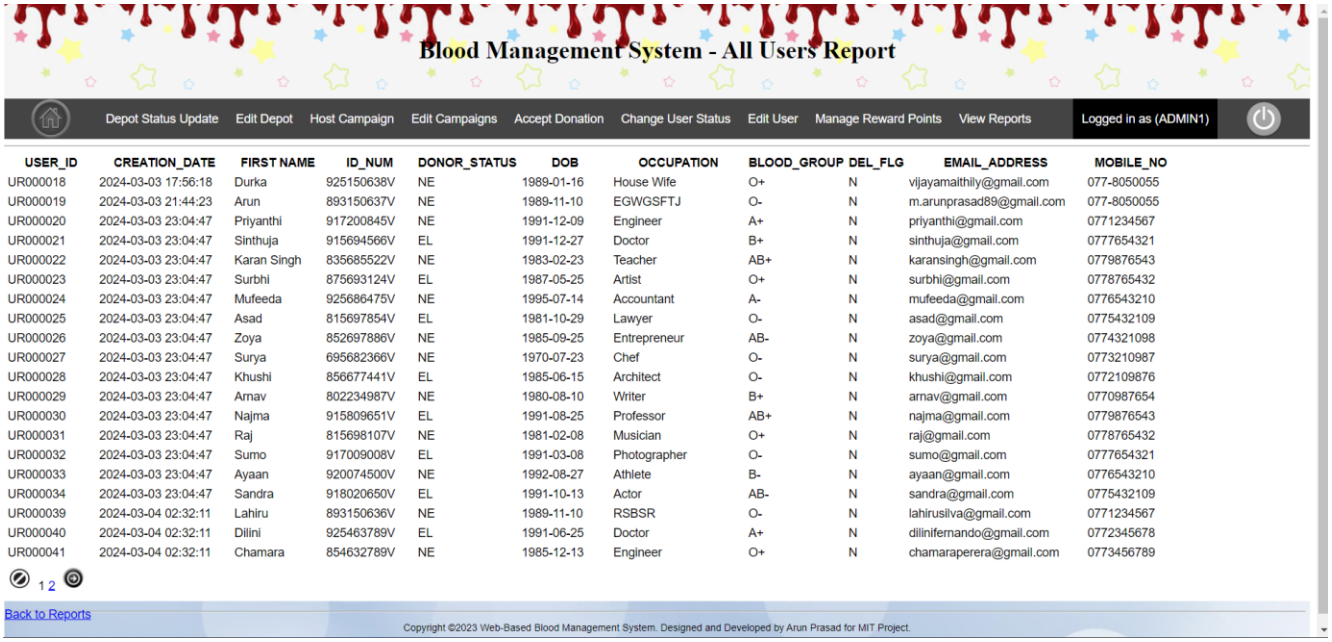


Figure D.1 Home Page - Reports

Sample Reports.

Following figures D.2 -D.4 depicts the All User, All Blood Depot, All Active Campaign reports



Blood Management System - All Users Report

Depot Status Update | Edit Depot | Host Campaign | Edit Campaigns | Accept Donation | Change User Status | Edit User | Manage Reward Points | View Reports | Logged in as (ADMIN1)

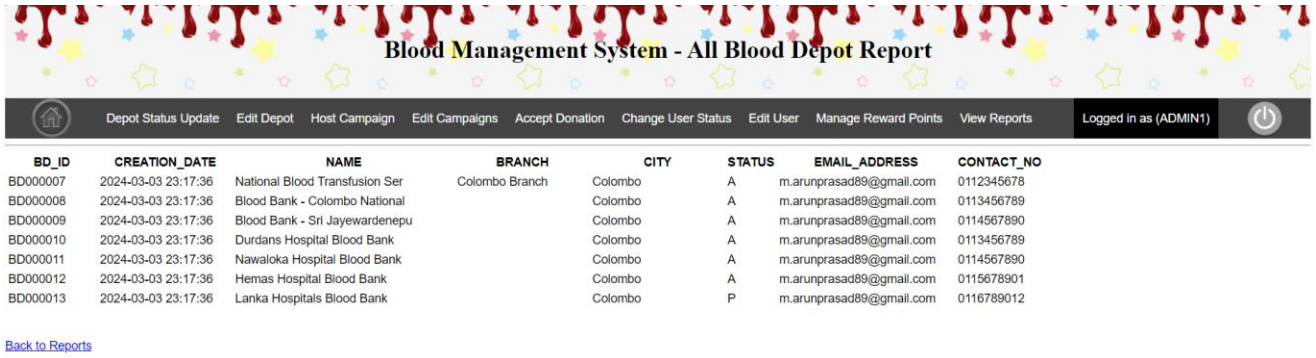
USER_ID	CREATION_DATE	FIRST NAME	ID_NUM	DONOR_STATUS	DOB	OCCUPATION	BLOOD_GROUP	DEL_FLG	EMAIL_ADDRESS	MOBILE_NO
UR000018	2024-03-03 17:56:18	Durka	925150638V	NE	1989-01-16	House Wife	O+	N	vijayamalthily@gmail.com	077-8050055
UR000019	2024-03-03 21:44:23	Arun	893150637V	NE	1989-11-10	EGWGSFTJ	O-	N	m.arunprasad89@gmail.com	077-8050055
UR000020	2024-03-03 23:04:47	Priyanthi	917200845V	NE	1991-12-09	Engineer	A+	N	priyanthi@gmail.com	0771234567
UR000021	2024-03-03 23:04:47	Sinthuja	915694566V	EL	1991-12-27	Doctor	B+	N	sinthuja@gmail.com	0777654321
UR000022	2024-03-03 23:04:47	Karan Singh	835685522V	NE	1983-02-23	Teacher	AB+	N	karansingh@gmail.com	0779876543
UR000023	2024-03-03 23:04:47	Surbhi	875693124V	EL	1987-05-25	Artist	O+	N	surbhi@gmail.com	0778765432
UR000024	2024-03-03 23:04:47	Mufeeda	925686475V	NE	1995-07-14	Accountant	A-	N	mufeeda@gmail.com	0776543210
UR000025	2024-03-03 23:04:47	Asad	815697854V	EL	1981-10-29	Lawyer	O-	N	asad@gmail.com	0775432109
UR000026	2024-03-03 23:04:47	Zoya	852697886V	NE	1985-09-25	Entrepreneur	AB-	N	zoya@gmail.com	0774321098
UR000027	2024-03-03 23:04:47	Surya	695682366V	NE	1970-07-23	Chef	O-	N	surya@gmail.com	0773210987
UR000028	2024-03-03 23:04:47	Khushi	856677441V	EL	1985-06-15	Architect	O-	N	khushi@gmail.com	0772109876
UR000029	2024-03-03 23:04:47	Arnav	802234987V	NE	1980-08-10	Writer	B+	N	arnav@gmail.com	0770987654
UR000030	2024-03-03 23:04:47	Najma	915809651V	EL	1991-08-25	Professor	AB+	N	najma@gmail.com	0779876543
UR000031	2024-03-03 23:04:47	Raj	815698107V	NE	1981-02-08	Musician	O+	N	raj@gmail.com	0778765432
UR000032	2024-03-03 23:04:47	Sumo	917009008V	EL	1991-03-08	Photographer	O-	N	sumo@gmail.com	0777654321
UR000033	2024-03-03 23:04:47	Ayaan	920074500V	NE	1992-08-27	Athlete	B-	N	ayaan@gmail.com	0776543210
UR000034	2024-03-03 23:04:47	Sandra	918020650V	EL	1991-10-13	Actor	AB-	N	sandra@gmail.com	0775432109
UR000039	2024-03-04 02:32:11	Lahiru	893150636V	NE	1989-11-10	RSBSR	O-	N	lahirusilva@gmail.com	0771234567
UR000040	2024-03-04 02:32:11	Dilini	925463789V	EL	1991-06-25	Doctor	A+	N	dilinifernando@gmail.com	0772345678
UR000041	2024-03-04 02:32:11	Chamara	854632789V	NE	1985-12-13	Engineer	O+	N	chamaraperera@gmail.com	0773456789

1 2

Back to Reports

Copyright ©2023 Web-Based Blood Management System. Designed and Developed by Arun Prasad for MIT Project.

Figure D.2 All User Report



Blood Management System - All Blood Depot Report

Depot Status Update | Edit Depot | Host Campaign | Edit Campaigns | Accept Donation | Change User Status | Edit User | Manage Reward Points | View Reports | Logged in as (ADMIN1)

BD_ID	CREATION_DATE	NAME	BRANCH	CITY	STATUS	EMAIL_ADDRESS	CONTACT_NO
BD000007	2024-03-03 23:17:36	National Blood Transfusion Ser	Colombo Branch	Colombo	A	m.arunprasad89@gmail.com	0112345678
BD000008	2024-03-03 23:17:36	Blood Bank - Colombo National		Colombo	A	m.arunprasad89@gmail.com	0113456789
BD000009	2024-03-03 23:17:36	Blood Bank - Sri Jayewardenepu		Colombo	A	m.arunprasad89@gmail.com	0114567890
BD000010	2024-03-03 23:17:36	Durdans Hospital Blood Bank		Colombo	A	m.arunprasad89@gmail.com	0113456789
BD000011	2024-03-03 23:17:36	Nawaloka Hospital Blood Bank		Colombo	A	m.arunprasad89@gmail.com	0114567890
BD000012	2024-03-03 23:17:36	Hemas Hospital Blood Bank		Colombo	A	m.arunprasad89@gmail.com	0115678901
BD000013	2024-03-03 23:17:36	Lanka Hospitals Blood Bank		Colombo	P	m.arunprasad89@gmail.com	0116789012

Back to Reports

Figure D.3 All Blood Depot report

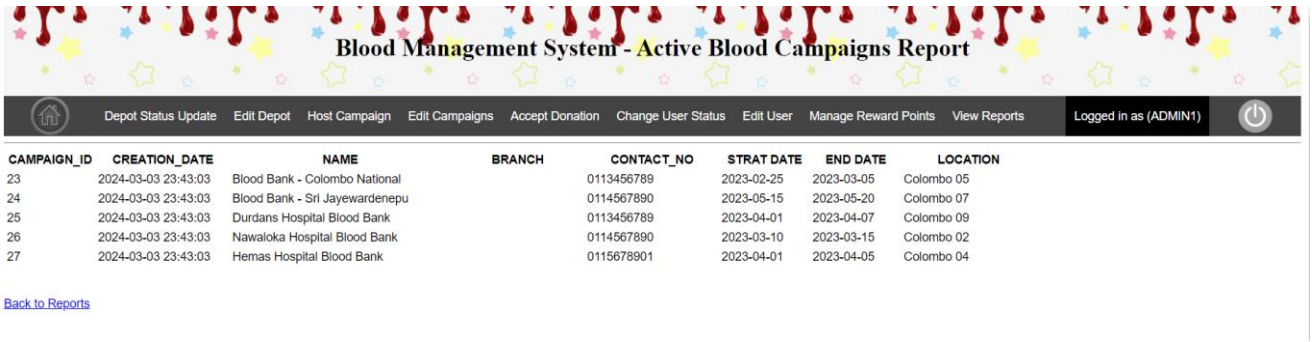


Figure D.4 All Active Blood Campaigns report

Below mentioned figures D.5- D.9 shows the other Important Management Reports.

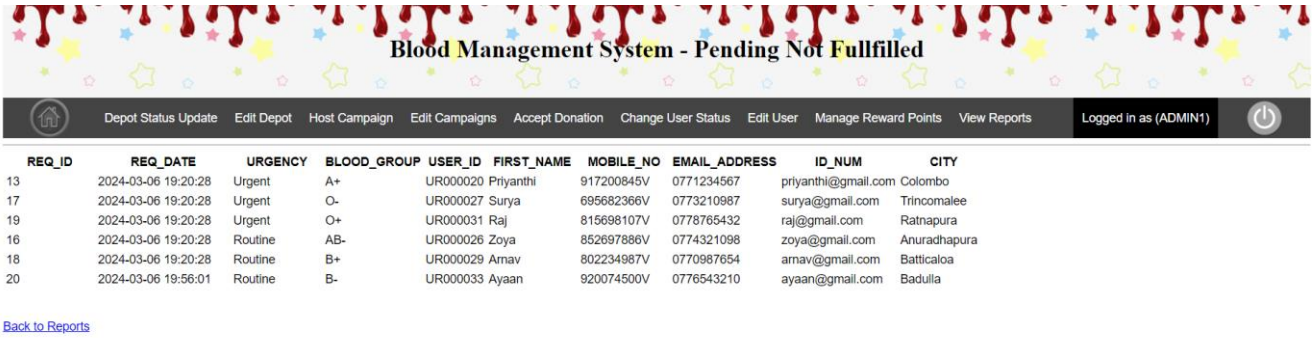
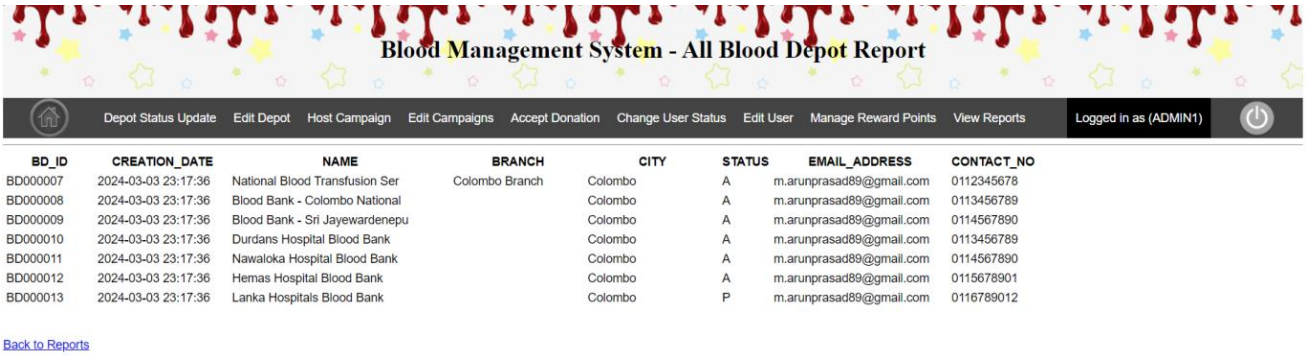


Figure D.5 Pending Not Fulfilled report

Figure D.6 Eligible Donor report

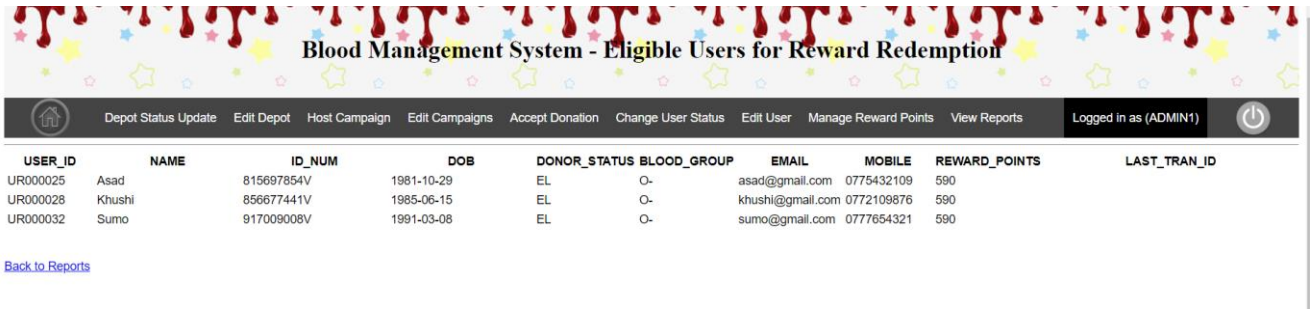


BD_ID	CREATION_DATE	NAME	BRANCH	CITY	STATUS	EMAIL_ADDRESS	CONTACT_NO
BD000007	2024-03-03 23:17:36	National Blood Transfusion Ser	Colombo Branch	Colombo	A	m.arunprasad89@gmail.com	0112345678
BD000008	2024-03-03 23:17:36	Blood Bank - Colombo National		Colombo	A	m.arunprasad89@gmail.com	0113456789
BD000009	2024-03-03 23:17:36	Blood Bank - Sri Jayewardenepu		Colombo	A	m.arunprasad89@gmail.com	0114567890
BD000010	2024-03-03 23:17:36	Durdans Hospital Blood Bank		Colombo	A	m.arunprasad89@gmail.com	0113456789
BD000011	2024-03-03 23:17:36	Nawaloka Hospital Blood Bank		Colombo	A	m.arunprasad89@gmail.com	0114567890
BD000012	2024-03-03 23:17:36	Hemas Hospital Blood Bank		Colombo	A	m.arunprasad89@gmail.com	0115678901
BD000013	2024-03-03 23:17:36	Lanka Hospitals Blood Bank		Colombo	P	m.arunprasad89@gmail.com	0116789012

Figure D.7 Donation Match report

TRAN_ID	TRAN_DATE	USER_ID	TRAN_TYPE	TRAN_AMT	TRAN_IND	REMARKS
11	2024-03-07 11:06:42	UR000018	SIGNUP	500.00	CR	Initial Topup
12	2024-03-07 11:06:42	UR000019	SIGNUP	500.00	CR	Initial Topup
13	2024-03-07 11:06:42	UR000024	SIGNUP	500.00	CR	Initial Topup
14	2024-03-07 11:06:42	UR000025	SIGNUP	500.00	CR	Initial Topup
15	2024-03-07 11:06:42	UR000028	SIGNUP	500.00	CR	Initial Topup
16	2024-03-07 11:06:42	UR000032	SIGNUP	500.00	CR	Initial Topup
17	2024-03-07 11:06:42	UR000025	DONATION	90.00	CR	Donation ID: 9
18	2024-03-07 11:06:42	UR000018	DONATION	100.00	DR	Donation ID: 9
19	2024-03-07 11:06:42	UR000028	DONATION	90.00	CR	Donation ID: 10
20	2024-03-07 11:06:42	UR000019	DONATION	100.00	DR	Donation ID: 10
21	2024-03-07 11:06:42	UR000032	DONATION	90.00	CR	Donation ID: 12
22	2024-03-07 11:06:42	UR000024	DONATION	100.00	DR	Donation ID: 12

Figure D.8 Transaction report



USER_ID	NAME	ID_NUM	DOB	DONOR_STATUS	BLOOD_GROUP	EMAIL	MOBILE	REWARD_POINTS	LAST_TRAN_ID
UR000025	Asad	815697854V	1981-10-29	EL	O-	asad@gmail.com	0775432109	590	
UR000028	Khushi	856677441V	1985-06-15	EL	O-	khushi@gmail.com	0772109876	590	
UR000032	Sumo	917009008V	1991-03-08	EL	O-	sumo@gmail.com	0777654321	590	

Figure D.9 Eligible Users for Reward Points Redemption report

APPENDIX E

Project Gannt Chart

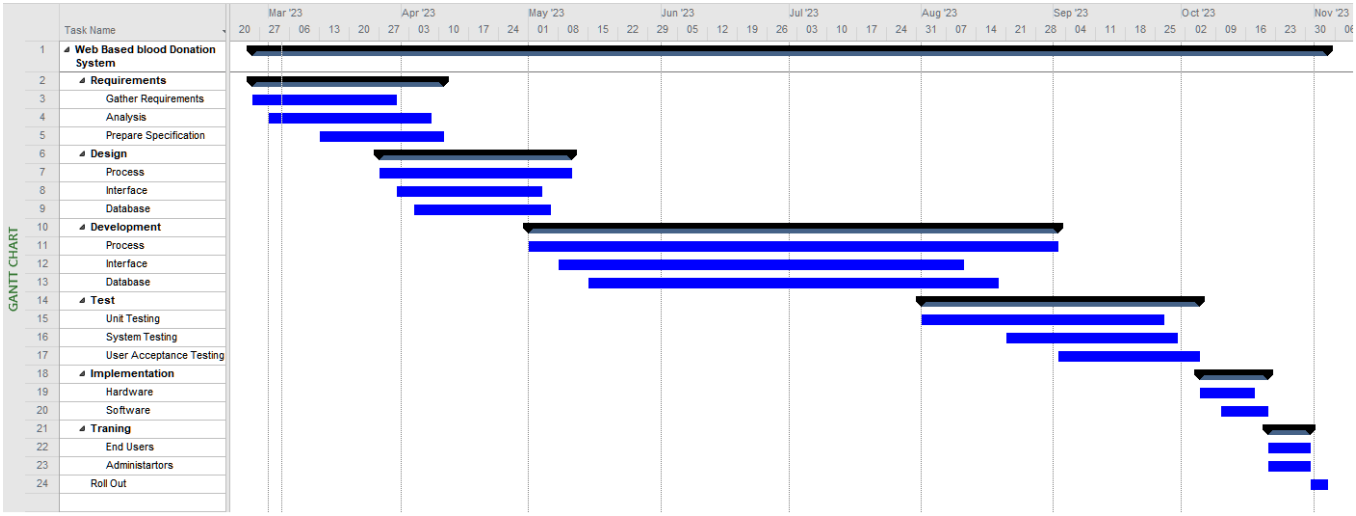


Figure E.1 The project's Gantt Chart illustrates the project's timeline and tasks.