 **Software Requirements Specification**

**Blood Bank Donation Management System**

**Submitted To**

**Khandoker Nosiba Arifin**

**Lecturer, Department of CSE, UITS**

**Submitted By**

**Team: RedRescue**

|  |  |  |
| --- | --- | --- |
| **No.** | **Name** | **ID** |
| 1 | Yeasmin Kabir Keya | 0432220005101107 |
| 2 | Arpa Bhowmik | 0432220005101102 |
| 3 | Sawda Akter | 0432220005101125 |
| 4 | Fahima Abida Chowdhury | 0432220005101135 |

**Contents**

**Section 1: Introduction**

## 1.1 Purpose

## 1.2 Scope

## 1.3 Intended Audience

## 1.4 Definitions, Acronyms and Abbreviations

# 2. Overall Description

## 2.1 Product Perspective

## 2.2 Product Functions

## 2.3 User Classes and Characteristics

## 2.4 Operating Environment

## 2.5 Design and Implementation Constraints

## 2.6 Assumptions and Dependencies

**3. Specific Requirements**

**3.1 Functional Requirements**

**3.2 Non-Functional Requirements**

**3.3 Interface Requirements**

**4. References**

**6. Appendices**

# Blood Bank & Donor Management System (BBDMS)

# 1. Introduction

## 1.1 Purpose

The purpose of the Blood Bank & Donor Management System (BBDMS) is to digitize and simplify the management of blood donor data and blood request processes. The system aims to connect blood donors with recipients through a centralized platform and help blood banks manage donor information efficiently. It provides functionalities for donor registration, blood search based on blood group and location, donor availability management, and a streamlined process for hospitals and individuals to request blood donations. The system is especially beneficial during emergencies, ensuring timely access to potential donors.

## 1.2 Scope

The BBDMS project is a full-stack web application designed to manage a blood donation system effectively. The system will provide modules for the following:  
- Donor registration and profile management.  
- Secure login and role-based access for users and admins.  
- Search functionality for locating blood donors by blood group and city.  
- Blood request posting and management by both users and administrators.  
- Admin dashboard to manage donors, blood groups, and request statuses.  
- Contact and feedback functionalities for better communication.  
  
The system is accessible via a web browser and is designed for use by blood banks, donors, and patients/recipients. The platform allows real-time updates, data validation, and user-friendly interfaces for effective system usage.

## 1.3 Intended Audience

This document is intended for the following stakeholders:  
- **Project Developers:** To understand the system requirements and functionalities.  
- **System Administrator:** To configure, maintain, and update the system.  
- **Academic Instructors and Evaluators:** To assess the design and implementation for academic purposes.  
- **End Users:** Including both donors and blood recipients who will interact with the system.  
**- Organizations:** Hospitals and NGOs managing blood donation drives.

## 1.4 Definitions, Acronyms and Abbreviations

**- BBDMS:** Blood Bank & Donor Management System  
**- Admin:** System user with privileges to manage the database and users.  
**- Donor:** A registered user willing to donate blood.  
**- Recipient/Requester:** An individual or hospital submitting a blood request.  
**- UI:** User Interface  
**- DBMS:** Database Management System

# 2. Overall Description

## 2.1 Product Perspective

This is a self-contained system developed using PHP for backend development and MySQL as the relational database. It is a web-based application compatible with popular browsers like Chrome and Firefox. The system is modular, separating the frontend user interface, business logic, and data access layers. The application can be deployed on both localhost (via XAMPP/WAMP) or a live hosting environment.

## 2.2 Product Functions

Key product functions include:  
- Donor registration and authentication.  
- Donor profile editing, availability status update.  
- Search donors by location and blood group.  
- Submit and view blood requests.  
- Admin panel for managing users and requests.  
- System-generated donor list and request records.  
- Contact and about us pages for organizational transparency.

## 2.3 User Classes and Characteristics

**- Admin:** Can add/edit/delete donor profiles, view and approve/reject blood requests, manage blood group database.  
**- Donor:** Can register, log in, view/edit profile, check blood requests.  
**- Guest/Visitor:** Can search for donors, contact the blood bank, but cannot request blood or access sensitive data.

## 2.4 Operating Environment

**- Frontend:** HTML, CSS, JavaScript, Bootstrap  
**- Backend:** PHP 7.x or later  
**- Database:** MySQL 5.7+  
**- Server:** Apache (localhost or cloud-hosted LAMP environment)  
**- Browser Support:** Chrome, Firefox, Edge, Safari

## 2.5 Design and Implementation Constraints

- System must be hosted on a PHP-supported server.  
- MySQL database configuration must be accurate.  
- User input validation is essential for data integrity.  
- Security measures must be implemented to prevent SQL injection and XSS attacks.

## 2.6 Assumptions and Dependencies

- All users will have internet access.  
- Donors provide accurate and truthful information.  
- Admin users have prior training in using the platform.  
- The system is maintained regularly to ensure donor data is up to date.

**3. Specific Requirements**

**3.1 Functional Requirements**

* **FR1**: The system shall allow new donors to register.
* **FR2**: The system shall enable searching of donors by blood type and location.
* **FR3**: Admin shall manage donor records.
* **FR4**: Admin shall approve/decline blood requests.
* **FR5**: The system shall allow users to request blood from specific donors.
* **FR6**: The system shall allow users to contact the admin via a contact form.

**3.2 Non-Functional Requirements**

**Performance:** The system should load pages and respond to user actions (like submitting a form or navigating between pages) within **3 seconds. It matters** to fast response times improve user experience and reduce frustration. Slow performance can drive users away.

**Security:** Instead of storing raw passwords in the database, the system uses cryptographic hashing (e.g., bcrypt or Argon2) to store a secure version of the password.If someone gains access to the database, they can't easily steal user passwords.

**Usability:** The system should be easy to navigate, with intuitive design, clear labels, and logical layout. Inputs in forms (e.g., email, password) should be checked for correctness both on the client side (JavaScript) and server side (PHP). It prevents errors, improves user feedback, and enhances security (e.g., avoiding SQL injection or malformed data).

**Maintainability:** Modular code, standard PHP practices

**3.3 Interface Requirements**

* **User Interface**: Intuitive and accessible via web browsers.
* **Database Interface**: MySQL database connected via PHP (PDO or MySQLi).
* **Hardware Interface**: Standard desktop/mobile hardware.

**4. References**

<https://www.w3schools.com>

<https://www.geeksforgeeks.org>

<https://www.php.net/docs.php>

<https://dev.mysql.com/doc/>

**6. Appendices**

* Appendix A: Entity Relationship Diagram (ERD)
* Appendix B: Screenshots of Interfaces
* Appendix C: Database Schema
* Appendix D: Class Diagram
* Appendix E: Use Case Diagram
* Appendix F: Activity Diagram