Software Design Specification

Blood Donation Management App

Project Code:

Internal Advisor:
Dr. Muhammad Ilyas

External Advisor:
Dr. Muhammad Ilyas

Project Manager:
Shumaila Zafar (BSAI51S25R028)

Project Team:
Rameen Fatima (BSAI51S25R039)
Zobia Shaheen (BSAI51S25R018)

Submission Date:
Dec. 10, 2025

_____
Project Manager's Signature

Document Information

Category	Information
Customer	UOS – Department of Computer Science
Project	Blood Donation Management System
Document	Software Design Specification
Document Version	1.0
Identifier	<PGBH01-2025-DS>
Status	Final
Author(s)	Rameen Fatima
Approver(s)	PM
Issue Date	Dec. 10, 2025
Document Location	UOS, Sargodha
Distribution	1. Advisor 2. PM 3. Project Office

Definition of Terms, Acronyms and Abbreviations

Term	Description
BDMS	Blood Donation Management System
DD	Design Specification
SRS	Software Requirements Specification
OTP	One Time Password
UI	User Interface
UML	Unified Modeling Language
DFD	Data Flow Diagram

Table of Contents

1. Introduction

1.1 Purpose of Document

- *This Document lays out the **design-level view** of the Blood Donation Management App. It describes how the system will be structured, what modules and classes will be built, and how data will flow through the app. The primary readers are the instructors, project manager, developer-students, and evaluators.*

- ***Design Methodology:***
  *We're using **Object-Oriented Design** because the project is built with OOP concepts, classes, inheritance, polymorphism, composition and aggregation.*

1.2 Project Overview

The Blood Donation Management App is an application used for managing:

- *User registration/login, donor/recipient/admin.*
- *Donor availability and appointment scheduling.*
- *Blood requests and status tracking.*
- *Stock management and reporting (blood bank/admin).*

*It reduces manual record keeping and enhances the facility of finding donors and handling requests. Design supports the key actors shown in the diagrams: **Donor, Recipient/Patient, Admin, Blood Bank**.*

1.3 Scope

What the system will do

- *Register/login users (donor/recipient/admin).*
- *Store Donor profile (blood group, age, location, donation history)*
- *Allow donors to update availability.*
- *Allow recipients to request blood and check the request status.*
- *Allow the admins to verify users and manage requests/users.*
- *Allow the blood bank/admin to manage blood stock and generate reports.*

What the system will not do

- *Real time integration with hospital hardware systems.*
- *Actual online payment or tracking of delivery.*
- *Full integration with Google Maps.*
- *Advanced chat system between donor and receiver.*

2. Design Considerations

2.1 Assumptions and Dependencies

The design respects the assumptions in the SRS, such as:

- *Must have stable internet connection.*
- *OTP verification/ Email verification supported.*
- *Location permissions are expected.*

In designs term, this leads to:

- *Confirmation at every input point.*
- *Role-based access (donor/recipient/admin).*
- *Optional mock services for OTP during the tests.*

2.2 Risks and Volatile Areas

Main risks:

1. ***Incorrect user data (****fake profiles, wrong blood group)*
    - *Solution: Admin verification and account status flags.*
2. ***Security concerns*** *(password handling, privacy)*
    - *Solution: Basic encryption/hashing approach at design level and OTP step.*
3. ***Requirements change*** *(new fields, reports)*
    - *Solution: Modular design with distinct modules for users, requests and stocks.*
4. ***Limited hardware/platform support***
    - *Solution: Keep UI simple (menu driven) and keep portable code.*

3. System Architecture

The Blood Donation Management System has been divided into simple components, so each part of the system will play a distinct role. Major components include:

- *The menu-driven system will employ a console-based menu.*

Functional Modules:

- ***User management:*** *Registration/Login/Verification.*
- ***Donor Module:*** *Availability, Donation details.*
- ***Recipient/Patient Module:*** *Request Blood, Check Status.*
- ***Blood Bank Module:*** *Stock update, View stock.*
- ***Appointment Module:*** *Schedule and view appointments.*
- ***Report Module:*** *Generation of basic reports*

Storage Module*: File-based storage of records, like users, requests, stock and appointments.*

All the parts work together to facilitate user registration, blood requests and management of stocks.

3.1 System Level Architecture

- *It is divided into three major parts: Namely UI, Logic Modules, and Storage.*
- *These components communicate by simple function calls.*
- *External users of the system use it via a console UI: Donor, Recipient, Admin and Blood Bank.*
- *It will involve all components residing on one computer: a single-machine C++ application.*
- *The following will handle all errors:*
    - *Input validation, checking for wrong entries.*
    - *Safe menu handling-no crashing.*
    - *Clear error message.*

3.2 Sub-System / Component / Module Level Architecture

The major modules of the Blood Donation Management System include:

- ***User Management module:***
  Handles:
  - *Registration/Login*
  - *OTP simulation, if needed.*
  - *User roles (Donor/Recipient/Admin).*
  - *Admin verification Status*
- ***Donor Module:***
  Handles:
  - *Donor profile: blood group, age, location.*
  - *Mark as available/unavailable.*
  - *View Donation History*
  - *Appointment status view.*
- ***Patient/Recipient Module:***
  Handles:
  - *Create blood request.*
  - *Check request status.*
  - *Cancel request.*
- ***Blood Bank Module:***
  Handles:
  - *View blood stock.*
  - *Add/update blood units.*
  - *Check the availability based on blood group.*
- ***Appointment Module:***
  Handles:
  - *Schedule appointment donor date/time.*
  - *View future appointments.*
- ***Report Module:***
  Handles:
  - *Stock report: Blood group wise*
  - *Request report pending/accepted/completed*
  - *Verify users list.*

3.3 Sub-Component / Sub-Module Level Architecture (1…n)

- ***User Management Module****:*
  - *The registration handler takes user information and stores it.*
  - ***Login Validator:*** *Check emails/password.*
  - ***Role Manager****: Donor/Recipient/Admin menu access.*
  - ***Verification Handler:*** *Admins verify users.*
- ***Donor Module:***
  - *Profile Manager views/updates donor information.*
  - ***Availability Manager:*** *Update availability.*
  - *Appointment viewer (view appointment status).*
- ***Recipient Module:***
  - ***Request Creator****: Creates a new request for blood.*
  - ***Request Tracker:*** *Display statue.*
  - ***Request Canceller:*** *Cancel a request.*
- ***Blood Bank Module****:*
  - ***Stock Viewer:*** *Display blood group and available units.*
  - ***Stock Updater:*** *Add/remove units.*
  - ***Optional Warning:*** *Low stock checker*

- ***Appointment Module:***
  - ***Appointment Scheduler:*** *Schedule donation time.*
  - ***Appointment Manager:*** *Cancel/Reschedule.*
- ***Report Module****:*
  - *Stock report generator.*
  - *Request report generator.*
  - *Optional user verification report.*

4. Design Strategies

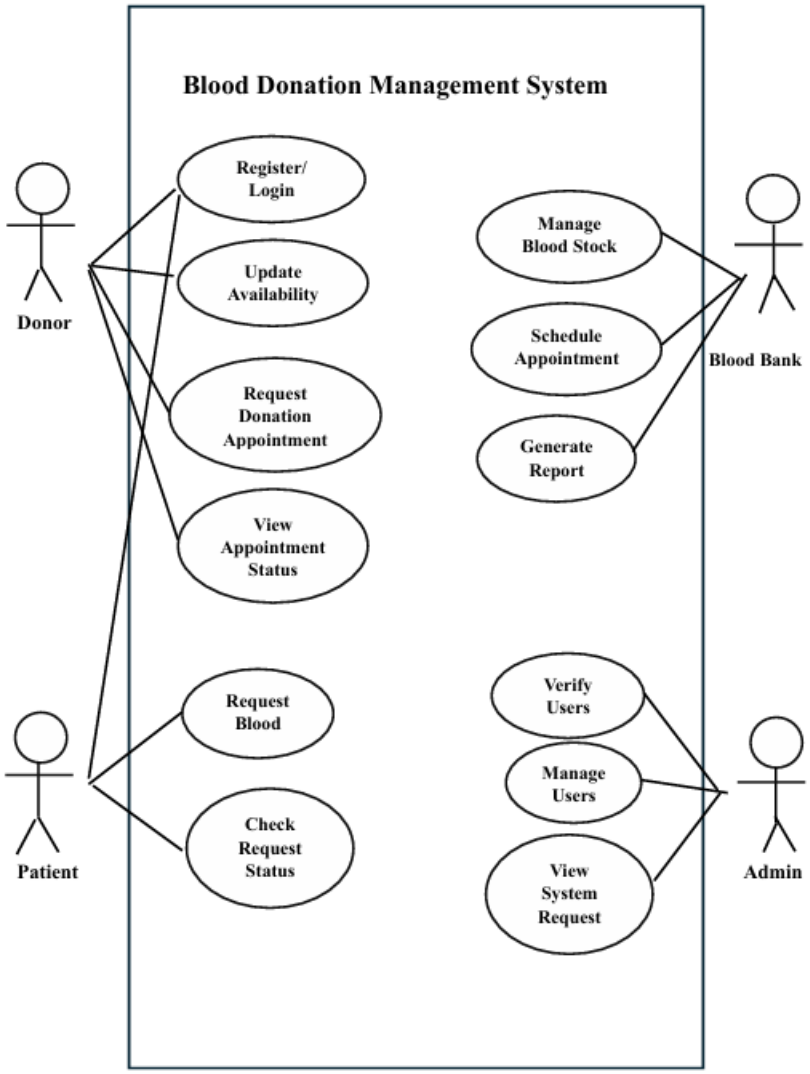
- ***Design in modular:*** *The system is further divided into modules like User, Donor, Recipient, Stock, Reports.*
- ***Object-oriented approach:*** *This makes the code reuseable because it structures the code in classes using OOP concepts.*
- ***Simple Console UI****: Menu-based UI, it works on most computers easily.*
- ***File-Based Storage****: Records are stored in files, not in some complex database.*
- ***Error Handling****: Input Validation, handling wrong choices, avoiding crashes.*
- ***Security****: Password are stored carefully at least not shown in output: Admin's verification prevents the creation of fake users.*

4.1 Strategy 1…n

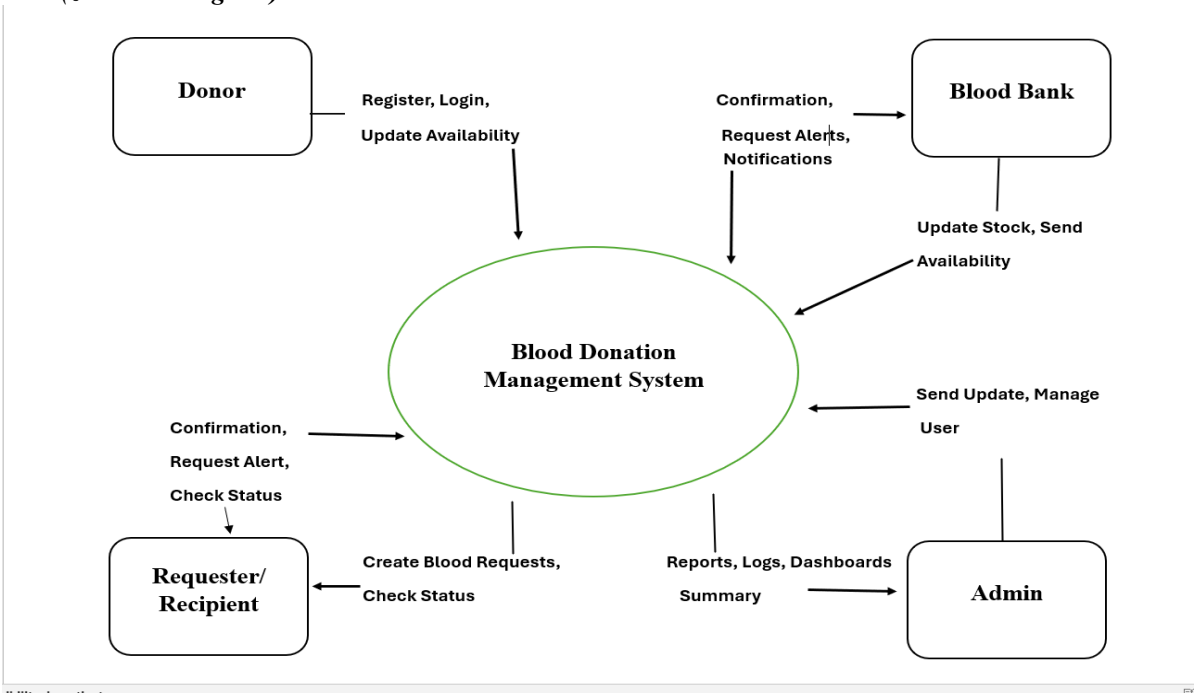
- ***Future Extension***
  Modular design will also allow for easy addition of future features, including searching for donors by city and sending out emergency alerts.
- ***System Reuse***
  OOP helps reuse the common features such as login, menu display, input checking.
- ***User Interface***
  Console UI is easy and simple, but it won't look like a mobile app, which is totally dine for the C++.
- ***Data Management***
  The implementation of file-based records is easy; however, judicious handling is required to avoid duplication or wrong data.
- ***Reliability***
  Validation and clear status tracking provides stability to the system.

5. Detailed System Design

- ***Use Case Diagram:***



- ***DFD (zero level diagram):***



6. References

- *Course Instructor guidelines*
- *Lecture Notes*
- *Google*
- *Seniors*
- *Civil Hospital Sargodha*