Software Requirements Specification (SRS)

Blood Management System (BMS)

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) is to define and document all functional and non-functional requirements of the Blood Management System (BMS). This document ensures clear communication among developers, stakeholders, and end-users regarding system expectations, behavior, and constraints.

The system is designed for hospitals, clinics, and blood banks to manage donor data, patient data, blood stock, and blood assignment history with Excel-based persistent storage.

1.2 Scope

The Blood Management System (BMS) is a Python-based application that performs:

- Donor registration
- Patient registration
- Determination of patient blood needs via conditional logic
- Blood assignment with stock validation
- Real-time blood stock tracking
- Viewing donor, patient, stock, and assignment history

- File handling with Excel integration for permanent data storage

The system ensures that existing records are never overwritten or deleted, and all new data is appended. Excel acts as the primary database backend.

1.3 Definitions, Acronyms, and Abbreviations

Term	Meaning
BMS	Blood Management System
Excel DB	Logical database implemented using Excel files
Donor	Person who donates blood
Patient	Person potentially receiving blood
Unit	A single blood donation measure
Stock	Current count of each blood group in inventory

1.4 References

- IEEE SRS Standard 830-1998
- Python 3.x Documentation
- openpyxl Library Documentation
- Healthcare Blood Storage Guidelines (WHO Standards)

2. Overall Description

2.1 Product Perspective

BMS is a standalone application with modular architecture. The system interacts exclusively with Excel files for data persistence. It replaces manual paper-based systems and legacy spreadsheets with an automated, controlled, and validated workflow.

2.2 Product Features

1. Add Donor
2. Add Patient
3. Determine if patient needs blood
4. Assign blood with stock validation
5. View Donor List
6. View Patient List
7. View Blood Stock
8. View Blood Assignment History
9. Auto-save and auto-update Excel database

2.3 User Classes and Characteristics

User Class	Description
Administrator	Operates full system, manages donors, patients, and stock
Medical Staff	Enters patient needs and checks stock
Record-Keeping Staff	Reviews reports, ensures compliance

2.4 Operating Environment

- Python 3.x
- Windows / Linux / macOS environment
- Required library: openpyxl
- Excel 2010+ compatible .xlsx format

2.5 Design and Implementation Constraints

- Excel is the only storage backend
- Non-destructive data handling (cannot delete previous data)
- Local file system dependency
- No GUI (Command-Line Interface only)

2.6 Assumptions and Dependencies

Assumptions

- Users provide correct blood group information.
- Correct formatting of Excel files remains intact.
- The system runs locally, not online.

Dependencies

- Python environment stability
- Excel compatibility
- openpyxl read/write capabilities

3. System Features (Functional Requirements)

3.1 Add Donor

Description

The system captures donor information and updates the donor database and blood stock.

Functional Requirements

FR-1.1: System shall accept donor name, age, blood group, and contact.

FR-1.2: System shall validate blood groups.

FR-1.3: System shall append donor data to Excel file.

FR-1.4: System shall update blood stock for the donated blood group.

FR-1.5: System shall not delete or overwrite previous donor records.

3.2 Add Patient

Description

The system records patient data and checks if the patient requires blood.

Functional Requirements

FR-2.1: System shall accept patient name, age, and remarks.

FR-2.2: System shall ask: "Does the patient need blood? (yes/no)"

FR-2.3: If yes, the system shall require blood group input.

FR-2.4: System shall call blood assignment logic if blood is needed.

FR-2.5: System shall append patient records to Excel without overwriting.

3.3 View All Donors

FR-3.1: System shall fetch donor data from Excel.

FR-3.2: System shall display all donor entries.

FR-3.3: System shall not modify donor data during viewing.

3.4 View All Patients

FR-4.1: System shall fetch patient data from Excel.

FR-4.2: System shall display patient list.

3.5 View Blood Stock

FR-5.1: System shall fetch current stock from Excel.

FR-5.2: System shall present stock counts for each blood group.

FR-5.3: System shall only update stock when donors donate or patients receive blood.

3.6 Assign Blood to Patient

Description

System validates stock and logs to the assignment.

Functional Requirements

FR-6.1: System shall confirm stock availability before assigning blood.

FR-6.2: If stock is insufficient, the system shall reject the request.

FR-6.3: System shall decrement stock upon assignment.

FR-6.4: System shall log assignment details in Excel.

FR-6.5: System shall timestamp each assignment.

3.7 View Blood Assignment History

FR-7.1: System shall display a complete history of all blood assignments.

FR-7.2: System shall retain all previous logs permanently.

4. External Interface Requirements

4.1 User Interface

- Command-line menu
- Step-by-step input prompts
- Error messages for invalid input
- Menu loops until exit

4.2 Hardware Interface

- Standard keyboard
- Standard display monitor

4.3 Software Interface

- Uses openpyxl to interact with Excel files
- Local .xlsx files act as data warehouse

4.4 Communications Interface

- No network communication required

5. Non-Functional Requirements (NFRs)

5.1 Performance Requirements

NFR-1: System shall add or retrieve data within 1 second for each operation.

NFR-2: Excel interactions shall remain optimized via openpyxl.

5.2 Safety Requirements

NFR-2.1: System shall prevent blood assignment if stock is zero.

NFR-2.2: System shall avoid data loss during file operations.

5.3 Security Requirements

NFR-3.1: System shall prevent unauthorized modification of Excel files.

NFR-3.2: No password handling required since the system is offline.

5.4 Reliability

NFR-4.1: Excel files shall remain readable even after abrupt shutdowns.

5.5 Maintainability

NFR-5.1: Code shall be modular to support upgrades.

NFR-5.2: Data schema shall support expansion (e.g., adding more fields).

5.6 Usability

NFR-6.1: Interface shall be easy for non-technical hospital staff.

6. Data Requirements

6.1 Donor Data Fields

- Donor ID
- Name
- Age
- Blood Group
- Contact
- Donation Date

6.2 Patient Data Fields

- Patient ID
- Name
- Age
- Need Blood (Yes/No)
- Required Blood Group (optional)
- Date

6.3 Stock Data Fields

- Blood Group
- Units Available

6.4 Assignment History Fields

- Assignment ID
- Patient Name
- Blood Group
- Units Assigned
- Date/Time

7. System Architecture

7.1 Components

- Donor Module
- Patient Module
- Blood Stock Module
- History Module
- File Handling & Excel Database Module

7.2 Data Flow

1. Input received via CLI
2. Processing via core logic
3. Output to:
    a. Console
    b. Excel files

8. Constraints

- Excel file corruption risk
- Excel capacity limitations

- Local-only storage (no cloud backup)

9. Future Enhancements

- GUI with Tkinter or PyTQ
- Admin login system
- Automated email alerts for low stock
- Cloud database
- Donor eligibility checker

10. Conclusion

The Blood Management System is a robust, scalable, and practical tool for healthcare environments requiring a reliable blood record tracking solution. With Excel-based non-destructive storage, modular logic, and strict validation, the system ensures operational reliability and preserves all historical data.