Case Study: Blood Management System (BMS)

1. Executive Summary

The Blood Management System (BMS) is designed to modernize how hospitals record, manage, and track blood-related operations. The system provides robust data handling, donor–patient management, real-time blood stock monitoring, and automatic Excel-based storage that ensures historical integrity.

The solution emphasizes operational efficiency, accountability, and accuracy—key pillars in healthcare data management. With real-time updates, non-destructive file handling, and strict data persistence, the BMS reduces manual errors and supports informed decision-making.

2. Problem Statement

Hospitals traditionally rely on manual registers for recording donor information, patient blood needs, stock updates, and transfusion histories. This manual process results in:

- Data redundancy
- High risk of human error
- Difficulty retrieving historical records
- No real-time visibility into blood inventory
- Risk of running out of critical blood groups
- Loss of old data due to poor file management

The objective was to design a system that eliminates these shortfalls while keeping operations simple and user-friendly.

3. Project Objectives

The Blood Management System aims to:

1. Centralize and digitize donors and patient records.
2. Maintain real-time blood stock inventory.
3. Enable safe and controlled blood assignments to patients.
4. Automatically store all records in an Excel database without overwriting old data.
5. Provide a reliable, always-linked data backbone between the program and Excel.
6. Ensure operational continuity through automated saving, updating, and retrieval.

4. System Scope

The system covers:

- Donor Registration
- Patient Registration
- Determining if patient requires blood
- Assigning blood units
- Blood stock management
- Viewing donor/patient lists
- Viewing transfusion history
- Excel file integration for all persistent data storage

5. Functional Requirements

5.1 Donor Module

The system allows hospital administrators to:

- Add new donor information:
  - Name
  - Age
  - Blood Group
  - Contact Information
- Automatically update donor records in Excel.
- Maintain lifelong donor data without deletion or overwriting.

5.2 Patient Module

The system supports:

- Adding a new patient's information:
  - Name
  - Age
  - Health status
  - Blood requirement check (Yes/No)
- If patient needs blood:
  - System requests blood group
  - Checks blood stock availability
  - Assigns blood if stock is sufficient
  - Updates stock count
  - Logs the transaction into patient history

Excel auto-updates the following tables:

- Patients
- Blood Assignment History

5.3 Blood Stock Management

The system:

- Maintains inventory of blood groups
- Allows real-time stock viewing
- Updates stock automatically when:
    - Donor donates blood
    - Patient receives blood

The Excel sheet retains all previous stock data and adds every update.

5.4 Data Viewing

The system includes:

- View All Donors
- View All Patients
- View Blood Stock
- View All Blood Assignment History

All views fetch real-time data from the Excel file.

5.5 File Handling & Excel Integration

A core feature of the system is its non-destructive file handling:

- Excel stores all data permanently
- No old data is deleted
- Only new rows are appended
- Blood stock rows are updated, not overwritten elsewhere
- Program and Excel are always synchronized

6. System Architecture

The system operates on a modular architecture:

6.1 Layers

1. **User Interface Layer**

CLI-based menu system enabling:

   a. Donor management
   b. Patient management
   c. Blood stock view
   d. History view

2. **Logic/Processing Layer**

Contains business logic:

   a. Validating blood groups
   b. Checking stock availability
   c. Executing assignments
   d. Managing update workflows

3. **Data Layer**

Excel-based dataset:

   a. donors.xlsx
   b. patients.xlsx
   c. blood_stock.xlsx
   d. history.xlsx

Python libraries used:

- openpyxl — reading/writing Excel files
- os — file path handling
- datetime — timestamps

7. Workflow and System Operations

7.1 Adding a Donor

1. User selects “Add Donor”
2. System collects donor details
3. Validates blood group input
4. Adds donor entry to Excel
5. Increments blood stock count

7.2 Adding a Patient

1. User selects “Add Patient”
2. System collects patient details
3. Ask: "Does the patient need blood?"
4. If **No** → Patient record saved
5. If **Yes**:
    a. System asks for blood group
    b. Validates stock availability
    c. Deducts one unit
    d. Logs history in Excel

7.3 Viewing Data

Each view function pulls up-to-date data from Excel automatically.

7.4 Assignment Logging

Whenever a patient receives blood, system logs:

- Patient ID
- Donor (optional)
- Blood group
- Units assigned

- Date/time

8. Non-Functional Requirements

- **Reliability:** Excel storage ensures long-term data preservation.
- **Scalability:** New sheets/columns can be added without affecting the core logic.
- **Maintainability:** Modular architecture makes updates easier.
- **Performance:** Lightweight read/write operations keep execution fast.
- **Data Integrity:** No overwriting or loss of historical records.

9. Constraints

- System requires a Python and Excel-compatible environment.
- Manual cleanup may be needed if Excel files become excessively large.
- No cloud synchronization (local storage only).

10. Risks & Mitigation

Risk	Impact	Mitigation
Accidental deletion of Excel file	High	Keep backups; auto-create file if missing
Wrong blood group assignment	High	Strict validation checks
Data corruption due to abrupt closure	Medium	Transaction-based write operations
Stock miscount	Medium	Automatic increment/decrement logic

11. Key Outcomes

- Centralized blood management
- Significant reduction in human errors
- Faster decision-making regarding stock
- Complete history tracking
- Secure and persistent data storage
- Professional and operational-grade workflow

12. Conclusion

The Blood Management System is a purpose-built solution that modernizes hospital operations while retaining the simple, traditional workflow staff are accustomed to. Its live Excel integration ensures transparency and avoids data loss—two critical requirements in healthcare management.

This project demonstrates sound architecture, strong domain logic, and enterprise-level handling of file-based data. It is scalable enough to evolve into a full hospital information system in the future.