DBMS Project Synopsis: Blood Bank Management System

1. Title Page

- Project Title: Blood Bank Management System
- Course Name & Code: UCS310 – Database Management Systems
- Degree & Year: B.Tech (2nd Year)
- Department / Institute Name: [To Be Filled]
- Group Members (2–3 students) with Roll Numbers: [To Be Filled]
- Lab Instructor Name: [To Be Filled]
- Academic Year: 2025–2026

2. Introduction

The Blood Bank Management System addresses the need for a centralized database to track donors, blood donations, recipients, and blood inventory. Manual or file-based records make it difficult to maintain accurate stock levels, trace donation history, and respond quickly to urgent requests. A DBMS-backed solution improves reliability, availability, and integrity of critical healthcare data while enabling fast retrieval through SQL queries.

3. Problem Statement

Blood banks often struggle with fragmented data and delayed updates, leading to mismatches between available units and real demand. The existing manual processes lack strong validation, are error-prone, and provide limited auditability. This project proposes a structured database system that stores donors, donations, recipients, requests, and inventory in relational tables to ensure consistent, queryable, and up-to-date information.

4. Objectives of the Project

- To design the database using an ER model with clearly defined entities and relationships.
- To convert the ER model into relational tables with primary and foreign keys.
- To apply normalization (up to 3NF) to reduce redundancy and update anomalies.
- To implement the database using SQL DDL and DML.
- To enforce data consistency via constraints and transactional

updates.

5. Scope of the Project

- Functional boundaries: donor registration, donation logging, recipient registration, blood requests, and inventory tracking.
- Types of users: admin/staff responsible for data entry and verification.
- Backend-focused modules: donors, donations, recipients, requests, inventory.

6. Proposed System Description

The system records donor profiles and their donations, tracks recipient profiles and their blood requests, and maintains an inventory summary by blood type. Each donation increases inventory, while approved requests reduce inventory. The database schema supports fast lookup of available units, tracking of donation history, and status management for requests. Overall, the system improves efficiency, data consistency, and traceability of blood units.

7. Database Design

7.1 Entity–Relationship (ER) Diagram

Entities and relationships: - Donor (1) — (M) Donation - Recipient (1) — (M) Request - Inventory (by blood_type) linked to donations and requests Cardinality and constraints are enforced via foreign keys and unique constraints.

7.2 Relational Schema

- donors(id PK, name, blood_type, phone, city, created_at, deleted_at)
- donations(id PK, donor_id FK -> donors.id, blood_type, units, donation_date, expiry_date, deleted_at)
- recipients(id PK, name, blood_type, phone, hospital, created_at, deleted_at)
- requests(id PK, recipient_id FK -> recipients.id, blood_type, units, status, request_date, deleted_at)
- inventory(id PK, blood_type UNIQUE, units, deleted_at)

8. Normalization

Key functional dependencies: - donors: id → (name, blood_type, phone, city, created_at) - donations: id → (donor_id, blood_type, units, donation_date, expiry_date) - recipients: id → (name, blood_type, phone, hospital, created_at) - requests: id → (recipient_id, blood_type, units, status, request_date) - inventory: blood_type → units

All relations are in 3NF: attributes depend on the key, the whole key, and nothing but the key.

9. Database Implementation

9.1 SQL Implementation

- DDL: CREATE TABLE for donors, donations, recipients, requests, inventory; constraints for PK/FK/UNIQUE.
- DML: INSERT for new donors/recipients/donations/requests; UPDATE for inventory and request status; DELETE via soft deletes (`deleted_at`).
- Queries: joins between donors–donations and recipients–requests; aggregate functions for inventory reporting; filters by blood_type and date ranges.

9.2 PL/SQL Components

SQLite does not support PL/SQL stored procedures. Business logic is handled in the Go application layer; optional triggers can be added for inventory updates if required.

10. Transaction Management & Concurrency (Optional but Recommended)

- Use transactions for donation creation and inventory increments.
- Use transactions for request approval and inventory decrements.
- ACID properties are ensured by SQLite’s transactional guarantees.

11. Tools & Technologies Used

- DBMS: SQLite
- Backend: Go (`modernc.org/sqlite`)
- Frontend: HTML templates + CSS
- Interface tool (optional): SQLite DB Browser or command-line tools

12. Expected Outcomes

- A normalized database schema for blood bank operations.
- Reliable data retrieval using SQL joins and aggregates.
- Accurate inventory tracking aligned with donations and requests.
- Improved consistency and integrity through constraints and

transactions.