

**JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY**

**SCHOOL OF COMPUTING AND INFORMATION TECHNOLOGY**

**DEGREE: BSC. INFORMATION TECHNOLOGY**

**PROJECT TITLE: SMART BLOOD BANK MANAGEMENT SYSTEM(SBBMS)**

**STUDENT NAME: DAVID KYALO MBITHI**

**REGISTRATION NUMBER: SCT221-C004-0499/2021**

**SUPERVISOR: Dr. JUDY GATERI**

This project has been submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Science in Information Technology in the year 2025.

# **DECLARATION**

I affirm that the content and information presented in this document and program are entirely my own. If there is any borrowed information or content, I have duly provided proper references.

**NAME:** DAVID K. MBITHI

**REGISTRATION NUMBER:** SCT221-C004-0499/2021

**DATE:** 6/17/2025

**APPROVED BY:**

**Supervisor** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Date** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Sign** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# **ACKNOWLEDGEMENT**

I express my heartfelt gratitude to everyone who played a role in the successful completion of this project. Their unwavering support has been invaluable, guiding me from the project's conception through to its implementation. I am particularly grateful to my supervisor, Dr. Judy Gateri, whose guidance and insistence on adhering to due diligence and proper processes have been crucial in navigating this project. Her commitment to precision has been instrumental in avoiding potential errors and ensuring the project's success. Lastly, I extend my appreciation to my parents, wife, friends, and classmates who have accompanied me on this academic journey. Their encouragement has been a source of strength throughout my time in college.

# **DEDICATION**

I dedicate this project to the relentless pursuit of knowledge and the unwavering spirit of curiosity that fuels the journey of discovery. This endeavor is dedicated to those who believe in the power of learning and the transformative potential it holds. In heartfelt appreciation, I dedicate this project to my wife, whose encouragement has been a beacon of wisdom and support throughout the entire process. Her commitment to excellence has inspired me to strive for the highest standards in every aspect of this undertaking. Furthermore, I dedicate this project to my parents, whose boundless encouragement and unwavering belief in my abilities have been a constant source of motivation. To my friends and classmates, thank you for the camaraderie, shared experiences, and the collective strength that comes from learning together. May this project stand as a testament to the collaborative spirit of those who value education, curiosity, and the pursuit of knowledge.

# **LIST OF FIGURES**

[Figure 1. prototyping model diagram. 48](bookmark://_bookmark31)

[Figure 2. folder structure diagram. 55](bookmark://_bookmark32)

[Figure 3.accuracy visualization diagram 59](bookmark://_bookmark33)

[Figure 4. .donation data screenshot 64](bookmark://_bookmark34)

[Figure 5. transfusion data screenshot 64](bookmark://_bookmark35)

[Figure 6. liquid model working; screenshot 65](bookmark://_bookmark36)

[Figure 7. liqud model matrices screenshot 65](bookmark://_bookmark37)

[Figure 8. liquid model training screenshot 66](bookmark://_bookmark38)

[Figure 9. river model working screenshot 66](bookmark://_bookmark39)

[Figure 10.river model training and matrices screenshot 67](bookmark://_bookmark40)

# **TABLE OF CONTENTS**

[DECLARATION 2](#_Toc29199)

[ACKNOWLEDGEMENT 3](#_Toc16279)

[DEDICATION 4](#_Toc9267)

[LIST OF FIGURES 5](#_Toc25588)

[TABLE OF CONTENTS 6](#_Toc7266)

[ABSTRACT 13](#_Toc2743)

[CHAPTER 1: INTRODUCTION 14](#_Toc15561)

[1.0 Introduction 14](#_Toc9109)

[1.1 Background 15](#_Toc23449)

[1.2 Problem Statement 15](#_Toc1026)

[1.3 Research Questions 16](#_Toc6237)

[1.4 Objectives 16](#_Toc30136)

[1.41 General Objectives 16](#_Toc1237)

[1.42 Specific objectives 17](#_Toc16031)

[1.5 Justification 17](#_Toc1486)

[1.6 Project Scope 17](#_Toc5328)

[1.7 Limitations. 18](#_Toc13196)

[CHAPTER TWO: LITERATURE REVIEW 19](#_Toc18373)

[2.1 Introduction 19](#_Toc32452)

[2.2 Literature Review 20](#_Toc28661)

[2.3 Theoretical Framework 21](#_Toc17985)

[2.3.1 1. Theoretical Foundations 22](#_Toc16498)

[2.4 Historical Development of SBBMS 22](#_Toc10901)

[2.4.1 Early Theories in Blood Bank Management Systems 23](#_Toc12858)

[2.4.1.2 Foundations of SBBMS 24](#_Toc6290)

[2.4.1.3 Technological Advancements 25](#_Toc16224)

[2.4.1.4 Challenges and Complexity 27](#_Toc2149)

[2.5 Methodology Used in Previous Studies 28](#_Toc21563)

[2.5.1 Data Collection 29](#_Toc7800)

[2.5.2 Types of Data Collected 29](#_Toc3893)

[2.5.3 Data Collection Mechanisms 29](#_Toc1170)

[2.5.4 Datasets and Databases used 29](#_Toc27637)

[2.5.4.1 Types of Datasets Used 29](#_Toc17611)

[2.5.4.2 Databases Used in SBBMS Research 30](#_Toc7417)

[2.5.4.3 Demand Prediction Models 30](#_Toc32079)

[2.5.4.4 Data Collection Mechanisms for Demand Prediction 30](#_Toc11995)

[2.5.4.5 Algorithmic Techniques 31](#_Toc15108)

[2.5.4.6 Parameters in the SBBMS Algorithms 31](#_Toc10958)

[2.5.4.7 Types of Neural Networks for the SBBMS 32](#_Toc31374)

[2.5.4.8 Hybrid Models 33](#_Toc23078)

[2.5.4.9. Algorithms Used in SBBMS 33](#_Toc2631)

[2.5.4.9.1 Online Decision Tree Regression (River) 33](#_Toc6875)

[2.5.4.9.2 Multi-Task Liquid Neural Network (PyTorch) 34](#_Toc7460)

[2.5.4.9.3. Inventory Management (Rule-based) 34](#_Toc30060)

[2.6 Trends and Patterns in Smart Blood Bank Management Systems (SBBMS) 34](#_Toc7585)

[2.6.1 Advancements in SBBMS 35](#_Toc26250)

[2.6.1.2 Transfer Learning Strategies 36](#_Toc2343)

[2.6.1.3 Multi modal Fusion Approaches 37](#_Toc9076)

[2.6.1.4 Real-Time Processing and Edge Computing 38](#_Toc8078)

[2.6.1.5 Block-chain for Transparency and Security: 38](#_Toc5619)

[2.6.1.6 AI-Powered Chat-bots and Virtual Assistants: 39](#_Toc20379)

[2.6.1.7 Telemedicine and Remote Blood Management: 40](#_Toc30025)

[2.7 Gaps in the Literature 40](#_Toc20670)

[2.8 Conclusion 40](#_Toc19352)

[CHAPTER 3: RESEARCH DESIGN AND METHODOLOGY 44](#_Toc6990)

[3.1 Introduction 44](#_Toc29581)

[3.2 Research Design 44](#_Toc15384)

[3.2.1.1 Justification for the Mixed-Methods Approach 45](#_Toc14418)

[3.2.1.2 Benefits of the Mixed-Methods Approach 45](#_Toc9473)

[3.2.2 System Development Methodology 46](#_Toc21008)

[3.2.2.1 Four Phases of RAD 46](#_Toc27648)

[3.2.2.2 Application in SBBMS Development 47](#_Toc2955)

[3.3 Participants or Sample 48](#_Toc13000)

[3.3.1 Datasets used 49](#_Toc417)

[3.3.1.1 Blood Transfusion Service Center Dataset (KAGGLE): 49](#_Toc13230)

[3.3.1.1.0 Sample size and characteristics 49](#_Toc9908)

[3.3.1.1.1 Environment 49](#_Toc5156)

[3.3.1.1.3 Ethical considerations 49](#_Toc26814)

[3.3.1.1.4 Data collection 49](#_Toc732)

[3.3.1.2 Donation data Dataset: 49](#_Toc24750)

[3.3.1.2.0 Sample size and characteristics 50](#_Toc9957)

[3.3.1.2.1 Environment 50](#_Toc18599)

[3.3.1.2.3 Ethical considerations 50](#_Toc3041)

[3.3.1.2.4 Data collection 50](#_Toc30630)

[3.4.1 Ensuring Validity 50](#_Toc18220)

[3.4.2. Ensuring Reliability 50](#_Toc18054)

[3.5 Data Analysis 50](#_Toc19302)

[3.5.1 Data Analysis Methods 51](#_Toc3566)

[3.5.2 Machine Learning Algorithms 52](#_Toc1298)

[3.5.3 Content Analysis 53](#_Toc3299)

[3.5.3.1 Thematic Analysis 53](#_Toc29031)

[3.5.4 Coding and Categorization 53](#_Toc17632)

[3.5.5 Data Preprocessing 54](#_Toc10386)

[3.5.5.1 Data Standardization 54](#_Toc32663)

[3.6 Folder Structure 54](#_Toc24296)

[3.6.1 Data Analysis Process 55](#_Toc5063)

[3.6.2 Data Cleaning and Preparation 56](#_Toc15587)

[3.7 Hypothesis Testing 56](#_Toc27092)

[3.7.1 Predictive Modelling 56](#_Toc11236)

[3.7.2 Data Visualization 56](#_Toc1533)

[3.7.3 Data Presentation 57](#_Toc26117)

[3.7.4 Use of Statistical Analysis Outputs 57](#_Toc19813)

[3.7.5 Model Performance Metrics 57](#_Toc18823)

[3.7.6 Presentation of AI and ML Results 58](#_Toc9296)

[3.7.7 Use of Dashboard and Interactive Tools 58](#_Toc13467)

[3.8 Visualization of Accuracy 58](#_Toc14135)

[3.8.1 Feature Extraction 59](#_Toc5277)

[3.8.1.2 Confusion Matrices 59](#_Toc3498)

[3.8.2 Classification 59](#_Toc27640)

[3.8.2.1 Auto Learning 60](#_Toc15641)

[3.8.2.2 Implementation 60](#_Toc27721)

[3.8.2.3 Training the Model 60](#_Toc754)

[3.8.3 Contribution to Existing Literature 61](#_Toc28284)

[3.8.4 Research Quality 61](#_Toc2240)

[3.9 Ethical Considerations 61](#_Toc8442)

[3.9.1 Conclusion 62](#_Toc27181)

[CHAPTER 4: IMPLEMENTATION 64](#_Toc17441)

[4.1 Implementation(screenshots) 64](#_Toc5134)

[4.1.2 River-Based Online Learning Script 67](#_Toc3305)

[4.1.2.1 Overview 67](#_Toc28497)

[4.1.2.2 Dependencies 68](#_Toc31380)

[4.1.2.3 Data Ingestion & Feature Engineering 68](#_Toc32010)

[4.1.3 PyTorch-Based Liquid Neural Network & Inventory System 69](#_Toc5595)

[4.3 Language Used 72](#_Toc1019)

[4.3.1 (River online‐learning models): 72](#_Toc1109)

[4.3.2 (PyTorch LiquidNN + inventory system): 72](#_Toc1484)

[4.4 System Requirements 72](#_Toc4551)

[4.4.1 Operating System 72](#_Toc13541)

[4.4.2 Filesystem 72](#_Toc19916)

[4.4.3 Python Environment 72](#_Toc13823)

[4.4.3.1 core Python packages 72](#_Toc30624)

[4.4.3.2 Standard library; logging, os, sys, datetime, warnings. 72](#_Toc19268)

[4.5 Hardware Requirements 73](#_Toc12593)

[4.6 Requirement Specifications 73](#_Toc7567)

[4.6.1. River-Based Online Learning Model 73](#_Toc31729)

[4.6.1.0 Functional Requirements 73](#_Toc15498)

[4.6.2.0 Non-Functional Requirements 74](#_Toc21753)

[4.6.3.0 PyTorch-Based Liquid Neural Network & Inventory System 75](#_Toc29869)

[4.6.3.1 Functional Requirements 75](#_Toc13179)

[4.6.3.2 Non-Functional Requirements 76](#_Toc21119)

[4.7 Conclusion 77](#_Toc924)

[Chapter 5: Conclusion 78](#_Toc9341)

[5.1 Summary of Findings 78](#_Toc18719)

[5.2 Contributions to Knowledge and Practice 79](#_Toc6547)

[5.3 Limitations 79](#_Toc12442)

[5.4 Future Research Direction 80](#_Toc18447)

[5.5 Final Remarks 80](#_Toc19199)

[REFERENCES 81](#_Toc2418)

[APPENDIX: 84](#_Toc20160)

[APPENDIX(A). RIVER MODEL CODE; 84](#_Toc14737)

[APPENDIX (B) LIQUID NEURAL NETWORK MODEL CODE: 92](#_Toc21217)

# **ABSTRACT**

This project presents an integrated system for blood donation and transfusion prediction, inventory management, and proactive notification for hospitals. By leveraging machine learning models—including both online learning with River and deep learning with PyTorch—the system forecasts daily blood donations and transfusions using historical donor and transfusion data, holiday effects, and flu indices.

The RIVER-BASED pipeline enables incremental learning and retraining on new verified data, ensuring sustained accuracy and adaptability to local shifts. The PyTorch-based neural model introduces a multitask "liquid neural network" architecture to jointly predict donation and transfusion demand, enabling efficient resource allocation.

The system further supports real-time, interactive prediction, inventory tracking across multiple hospitals, shelf-life management of blood units, and automated notifications for low stock scenarios—alerting both hospital management and donor groups. The combined framework is designed for hospital administrators, blood bank managers, and public health coordinators, aiming to optimize blood supply chains, minimize shortages, and enhance public health response through data-driven automation.

# **CHAPTER 1: INTRODUCTION**

## **1.0 Introduction**

Blood is a critical component of healthcare systems worldwide. It is required to replenish patients who suffer from lack of the same and accident victims, who most of the time have lost a lot of blood. Many hospitals have local blood banks which service their needs and are supplemented by blood donations and blood procurement from other blood banks. In Kenya, blood shortages have been a recurring issue, especially in emergency situations.

In a report by the auditor general, 46% of the total blood requested by hospitals in Kenya is not delivered. This is a big percentage considering blood is requested in times of fatal danger. Despite tremendous improvement in technology, hospitals still face challenges in acquiring adequate blood to service all their internal requests. This is brought about by inefficient inventory management, inability to predict blood demand, and delays in blood procurement. These shortages can lead to avoidable fatalities. An efficient blood supply management system should ensure that blood is available at the right time, right quantity, right type, and in the right place.

The current blood bank systems often rely on manual processes, making it difficult to; track, forecast demand, distribute and predict inventory levels accurately. This leads to blood wastage due to expiration and inability to respond promptly to urgent requests by doctors. Given the importance of ensuring a constant and reliable blood supply, there is need for an innovative solution that uses modern technology to improve the management of blood banks.

This project proposes the development of an AI-Powered smart Blood Bank System using THE LNN and ORM model, to enhance the efficiency and accuracy of blood supply management. By employing AI algorithms, this system will predict blood demand, optimize blood inventory management, and automate the blood distribution process. The system will facilitate real-time monitoring of blood stocks across various hospitals and blood donation centers. This seamless connection of stake holders will ensure proper blood management.

## **1.1 Background**

Blood banks play a vital role in healthcare. However, many of them face challenges such as; inefficient inventory management, unpredictable blood demand, blood expiration, and blood shortages, especially in emergencies. In Kenya, the manual processes which are often used result in delayed responses and blood wastage due to inaccurate tracking. A hospital could request blood from a blood bank which lacks the specific blood type, causing fatal delays, and maybe expiration of the said blood type in a blood bank where it is overstocked.

To address these issues, this project proposes the development of an AI-POWERED SMART BLOOD BANK SYSTEM using predictive algorithms and leveraged against LIQUID NEURAL NETWORKS(LNN). The choice of LNN is a requirement for a robust system that is intelligent. The system will perfect blood inventory management by forecasting demand, keeping an updated database, seamless stakeholders' connection and automating the distribution of blood across hospitals. This AI-driven solution aims to ensure that blood is available at the right place and at the right time, reducing blood wastage and saving lives.

## **1.2 Problem Statement**

The current blood bank management systems in many healthcare facilities remain manual or semi-automated. This results in significant inefficiencies that negatively affect the availability and management of blood supplies. One of the most critical issues is the unpredictability of blood demand. Without correct blood demand forecasting tools, healthcare facilities will always most likely experience blood shortages in emergencies or blood wastage due to overstocking. This imbalance between supply and demand compromises the ability to provide required blood types timely, and can lead to loss of life, especially in critical cases such as surgeries, accidents, or natural disasters. Additionally, blood wastage is another concern. Blood has a limited shelf life, and poor tracking of blood inventories often leads to blood expiring before it can be used hence increasing the shortage problem. This is due to inefficient inventory management systems that fail to; monitor stock levels in real time, predict blood expiration dates and forecast blood demand. Also, communication between hospitals, blood banks, and donors is still slow and inefficient, particularly in emergencies when rapid coordination is essential. This delay can result in critical time being lost, especially when quick blood transfusions are needed. A related challenge is the lack of proper blood donor engagement. Many potential donors are unaware of the immediate need for specific blood types or upcoming donation drives due to inadequate communication and advertisement efforts. The failure to maintain an active and informed donor base reduces turnout at donation drives, exacerbating blood supply issues. Addressing these inefficiencies through a more advanced and automated solution is essential to ensure that blood supplies are managed optimally, reducing wastage, and ensuring timely availability of blood during emergencies.

## **1.3 Research Questions**

1. How can artificial intelligence be used to accurately predict blood demand and refine inventory management in blood banks to minimize both blood shortages and wastage?
2. What are the most effective methods for integrating real-time communication between hospitals, blood banks, and donors to ensure timely responses during emergencies?
3. How can AI-driven systems improve donor engagement and participation in blood donation drives?

## **1.4 Objectives**

The AI-powered Smart Blood Bank Management System aims to address the outlined challenges by automating blood inventory management through AI algorithms that predict blood demand, monitor stock levels, and recommend optimal storage and distribution strategies, while also reducing wastage using smart expiration tracking based on the First-In-First-Out (FIFO) principle. It further enables data-driven decision-making by providing real-time analysis and reporting tools to hospital administrators and blood bank staff, improving resource allocation and facilitating coordination across hospitals, blood banks, and donors through a centralized platform. Additionally, the system fosters collaboration among stakeholders by enhancing donor engagement using AI-powered communication tools to promptly notify donors of urgent blood requirements and streamline the scheduling of timely donations.

### **1.41 General Objectives**

The general goal of the AI-powered Smart Blood Bank Management System is to enhance the efficiency of blood inventory management, reduce blood wastage, improve donor engagement, and ease real-time collaboration among stakeholders through AI-driven automation and data analysis.

### **1.42 Specific objectives**

1). To develop AI algorithms that accurately predict blood demand and transfusion requirements in blood banks.

2). To create a real-time communication system that integrates hospitals, blood banks, and donors.

3). To implement AI-driven tools that enhance blood inventory management.

## **1.5 Justification**

The need for an efficient and reliable blood bank management system is critical to saving lives in both routine and emergency healthcare situations. However, many healthcare facilities struggle with blood shortages and wastage due to outdated manual or semi-automated blood bank management systems. In Kenya, these challenges are worsened by limited resources, poor inventory tracking, and insufficient communication between blood banks, hospitals, and donors.

This research is being undertaken to address these issues by using artificial intelligence to automate and improve the entire blood management process. An AI-powered Smart Blood Bank Management System will provide a data-driven solution to; predict blood demand, minimize wastage, and streamline communication among stakeholders, ensuring that blood is available when and where it is most needed.

## **1.6 Project Scope**

The smart blood bank management system developed using LIQUID NEURAL NETWORK (LNN) will facilitate streamlined blood bank operations. Core functionalities include donor registration, inventory tracking, and automated critical level alerts. Hospitals can request blood through a centralized interface, triggering real-time inventory checks to expedite urgent fulfillment.

The system includes secure multi-user access, with tailored dashboards for donors, hospitals, and administrators, ensuring efficient role-based functionality. It integrates data analytics and machine learning for demand forecasting, aiding in proactive stock management. Designed with a focus on data security and regulatory compliance, this project aims to enhance efficiency, minimize wastage, and improve patient outcomes through optimized resource management.

## **1.7 Limitations.**

The smart blood bank management system faces several technical limitations. Its accuracy depends on consistent, real-time data input; incomplete or outdated entries may lead to errors in inventory predictions. Additionally, reliance on stable internet connectivity can hinder performance in low-connectivity areas, impacting timely request fulfillment.

Data security is another constraint, as handling sensitive donor information requires strict adherence to data protection standards, with ongoing work to maintain compliance. Scalability is also limited; higher user loads may demand server upgrades, increasing operational costs. Addressing these challenges is essential for system reliability and broader scalability.

# **CHAPTER TWO: LITERATURE REVIEW**

## **2.1 Introduction**

The increasing demand for efficient and reliable blood transfusion services has underscored the importance of developing advanced management systems in blood banks. Traditional methods of managing blood inventories, donor information, and distribution logistics often encounter challenges such as data inaccuracies, delays in blood provision, and inefficient donor-recipient matching. To address these issues, the integration of smart technologies into blood bank management systems has emerged as a promising solution (Kavulavu et al., 2022).

A Smart Blood Bank Management System (SBBMS) leverages modern technologies, including the Internet of Things (IoT), cloud computing, and artificial intelligence, to enhance the efficiency, accuracy, and responsiveness of blood bank operations. These systems facilitate real-time monitoring of blood inventories, streamline donor management, and optimize the distribution process, thereby improving overall healthcare delivery. For instance, IoT-based systems can monitor storage conditions and inventory levels in real-time, ensuring optimal preservation of blood products and timely replenishment of stocks (Alotaibi & Mehmood, et al., 2023). For example, cloud‑hosted platforms centralize donor, blood‑type, and stock data to provide stakeholders with up‑to‑the‑minute information and automated alerts for replenishment or temperature excursions (Mudunuri, Hullurappa, Vemula, & Selvakumar, et al., 2024).

The primary objective of this literature review is to critically examine existing SBBMS research, by analyzing how smart technologies such as machine learning forecasting modules, RFID tracking, and mobile‑app interfaces have been adopted in blood bank operations, evaluating their impact on operational efficiency and service quality, identifying technical, ethical, and infrastructural challenges, and uncovering gaps in current knowledge to guide future investigations (Ben Elmir, Hemmak, & Senouci et al., 2023).

By addressing these objectives, this review seeks to provide a comprehensive understanding of the current landscape of smart blood bank management innovations, including AI‑driven demand forecasting, IoT‑enabled cold‑chain monitoring, and blockchain‑based traceability and to highlight their transformative potential for reducing blood wastage, enhancing supply‑chain resilience, and improving patient outcomes across diverse healthcare environments (Subbiah et al., 2022).

## **2.2 Literature Review**

The integration of smart technologies into blood bank management systems has garnered significant attention in recent years, aiming to enhance the efficiency, accuracy, and reliability of blood donation and distribution processes. Maher et al. (2021) conducted a systematic review focusing on the enhancement of blood bank systems through modern technologies such as artificial intelligence (AI), the Internet of Things (IoT), blockchain, cloud computing, and machine learning. Their study categorizes existing research based on motivations, objectives, and challenges, providing a comprehensive overview of technological advancements in blood bank management

A smart platform-oriented approach aimed at creating a robust blood demand and supply chain was proposed by Alzubaidi et al. (2023). It suggested that employing machine learning and time series forecasting models, their system forecasted blood demand, classified donors, and scheduled donation appointments. This approach seeked to reduce uncertainty in blood demand, minimize wastage, and balance blood collection and distribution effectively.

Also, a systematic literature review to investigate mobile applications designed to track, attract, and retain blood donors, conducted by Alshurideh et al. (2023), included a pilot survey indicating a high willingness among younger individuals to use mobile apps for blood donation, suggesting the potential of mobile technology in enhancing donor engagement

The application of AI in blood management has been proposed to optimize inventory tracking and donor management. A study published in the International Research Journal of Engineering and Technology (IRJET) discusses AI-enabled smart blood management solutions that employ machine learning models to alert blood banks when forecasted demand is likely to surpass available supply, enabling proactive measures to prevent shortages (Kshirsagar & Phalle, 2023).

Despite these advancements, certain gaps and inconsistencies persist in the literature. While numerous studies have explored technological solutions for blood bank management, there is limited research on the integration of these technologies into existing healthcare infrastructures, particularly in resource‑limited settings (Barnes et al., 2022). Additionally, concerns related to data privacy, security, and ethical considerations in the deployment of smart blood bank systems require further investigation (Elnawawy et al., 2024).

Furthermore, the scalability and interoperability of these smart systems across different regions and healthcare systems remain underexplored, especially in low‑ and middle‑income contexts where fragmented infrastructures and limited resources hinder seamless integration (Jayathissa & Hewapathirana, 2023). The absence of standardized protocols and guidelines, such as HL7’s Fast Healthcare Interoperability Resources (FHIR) specifications or ISBT 128 labelling standards poses significant barriers to achieving widespread SBBMS adoption and operational effectiveness (Haug, Narus, Bledsoe, & Huff et al., 2018). While considerable progress has been made in developing prototype SBBMS frameworks, further research is necessary to address technical, ethical, and privacy challenges and to evaluate system performance in real‑world settings (Livieri, Mangina, Protopapadakis, & Panayiotou et al., 2025). Accordingly, this review focuses on strategies for integrating smart blood bank technologies into existing healthcare infrastructures, robust approaches to data privacy and security, and the development of comprehensive, standardized protocols to facilitate scalable and interoperable SBBMS implementations

## **2.3 Theoretical Framework**

The theoretical framework for the Smart Blood Bank Management System is based on the integration of information systems and healthcare management theories. The system applies principles from systems theory, which emphasizes the interaction of components such as blood donation, storage, and distribution to optimize resource management (Bertalanffy, 2020). It also utilizes the Technology Acceptance Model (TAM) to analyze user acceptance of new technological systems, highlighting factors like perceived ease of use and perceived usefulness (Venkatesh et al., 2021). Also, the framework incorporates data management theories that stress the importance of secure and efficient handling of data, which is critical in managing blood inventories and ensuring patient safety (Zhao et al., 2022). These theories aim to improve decision-making, reduce blood wastage, and enhance overall healthcare service delivery (Gupta et al., 2023).

#### **2.3.1 1. Theoretical Foundations**

Systems Theory, (von Bertalanffy et al., 2020) suggests that the SBBMS should function as an integrated system where each component, such as blood donations, storage, inventory management, and distribution, interacts to ensure smooth operations. Additionally, Technology Acceptance Model (TAM), (Venkatesh et al., 2021) helps guide the development of user-friendly interfaces and promotes the acceptance of the system by healthcare professionals. Data Management Theories, (Zhao et al., 2022) ensure that data is securely stored and processed in compliance with healthcare regulations. Finally, DSS and SCM theories, (Gupta et al., 2023), help design algorithms for decision-making and optimize the logistics and distribution of blood products.

By leveraging these theories, the SBBMS aims to improve the efficiency, security, and usability of blood bank operations, ensuring it meets the needs of both users and patients (Rashid et al., 2021; Elnawawy et al., 2024).

## **2.4 Historical Development of SBBMS**

The development of Smart Blood Bank Management Systems (SBBMS) has evolved significantly over time, with foundational systems relying on manual, paper‑based records and physical tracking of blood units,an approach that was inherently prone to human error, inefficiency, and challenges in handling large data volumes (Pravallika, Tharun, Satish Kumar, Sridevi, & Balaji et al., 2023).

In the 1990s and early 2000s, the first electronic blood bank management systems emerged, replacing paper ledgers with basic computerization that improved inventory control and streamlined donation tracking; however, these early digital systems still lacked real‑time monitoring capabilities, robust data analytics, and user‑friendly interfaces (Nzoka & Ananda et al., 2022).

With the advent of Internet of Things (IoT) sensors, cloud computing, and big‑data platforms over the past decade, modern SBBMS employ RFID tagging for continuous unit tracking, predictive analytics modules for demand forecasting, and decision‑support dashboards for optimal allocation. Transformations that have dramatically enhanced operational efficiency, safety, and system sustainability (Subbiah et al., 2022).

Today’s SBBMS architectures also integrate blockchain‑based traceability and AI‑driven supply‑chain optimization to ensure end‑to‑end transparency and to reduce wastage, underscoring how successive technological innovations have collectively revolutionized blood bank operations (Mudunuri, Hullurappa, Vemula, & Selvakumar et al., 2024)

### **2.4.1 Early Theories in Blood Bank Management Systems**

The earliest theories in blood bank management systems were grounded in basic logistical and inventory management practices aimed at ensuring blood products were available when needed while minimizing wastage. Foundational concepts such as Just‑in‑Time (JIT) and Materials Requirements Planning (MRP) laid the groundwork for later systems by focusing on waste reduction and optimized stock control (Balkhi, Alshahrani, & Khan et al., 2022). Another early theory was queuing theory, which was applied to model waiting times for blood donations and blood unit requests; this approach informed the design of initial patient wait‑list management and resource allocation protocols (Luo et al., 2020).

Subsequent developments drew on classical inventory models such as Economic Order Quantity (EOQ) and ABC classification to determine optimal reorder points and prioritize high‑value blood products. EOQ frameworks helped balance ordering costs against holding costs, reducing both stockouts and excess inventory, while ABC analysis segmented blood types by usage rate and criticality to focus management efforts where they were most needed (Ashraf, Haq, & Rehman et al., 2021).

Forecasting theories from time‑series analysis and machine learning further enhanced demand prediction for blood components. Models such as ARIMA and recurrent neural networks have been deployed to anticipate fluctuations in blood demand based on historical usage patterns and external factors like seasonal variations or public health emergencies (Lee & Chen et al., 2021).

Operations research contributed linear programming and network‑flow optimization techniques to SBBMS design, enabling the formulation of allocation problems as solvable mathematical programs; this allowed blood banks to optimize distribution routes, minimize transportation costs, and ensure timely delivery under capacity constraints (Kumar & Singh et al., 2022).

Quality‑management theories, including Lean management and Six Sigma, were adopted to streamline processes, reduce defects in handling and storage, and foster a culture of continuous improvement. Lean principles eliminated non‑value‑added activities in blood processing workflows, while Six Sigma tools identified and mitigated sources of variability in donor screening and product handling (Patel, Mehta, & Shah et al., 2020).

Finally, decision‑support frameworks such as Multi‑Criteria Decision Analysis (MCDA) and the Technology Acceptance Model (TAM) have been applied to assess trade‑offs among cost, safety, and accessibility, and to study stakeholder acceptance of smart blood bank technologies. MCDA techniques facilitate transparent, structured evaluation of competing objectives, whereas TAM helps predict and improve user adoption of new digital platforms (Wang & Zhang et al., 2021).

#### **2.4.1.2 Foundations of SBBMS**

Theoretical foundations for SBBMS have continued to broaden beyond classical inventory and queuing models to encompass; holistic, technology‑centered, and human‑focused perspectives. Systems theory underpins the view of a blood bank as an interconnected whole, where feedback loops and dynamic interactions among donors, inventory, and distribution channels are modeled to enhance system resilience and adaptability (Deepa & Kumar et al., 2021). Resilience engineering theory further complements this by emphasizing the capacity of SBBMS to anticipate, absorb, and recover from disruptions, such as sudden spikes in demand during emergencies, through robust design of redundancy and flexible resource allocation protocols (Patel & Reddy et al., 2022).

Advances in Cyber‑Physical Systems (CPS) theory frame modern SBBMS as integrations of computational algorithms and physical processes, wherein IoT sensors, actuators, and cloud analytics form a closed‑loop control environment that ensures precise environmental monitoring and automated corrective actions to maintain blood product integrity (Zhang & Xu et al., 2022). In parallel, Socio‑Technical Systems (STS) theory highlights the crucial interplay between technology and human actors, guiding the design of user‑centered interfaces and collaborative workflows that align AI recommendations with clinical decision‑making practices (Gonzalez & Patel et al., 2023).

From a behavioral standpoint, the Unified Theory of Acceptance and Use of Technology (UTAUT) and its extensions inform strategies to promote stakeholder adoption of SBBMS by considering factors such as performance expectancy, effort expectancy, social influence, and facilitating conditions. These models have been empirically validated in healthcare settings to forecast technology uptake and tailor training interventions (Singh, Sharma, & Verma et al., 2021). Socio‑Cognitive theory also contributes by modeling how cognitive perceptions, such as perceived control over blood safety, shape user engagement with decision‑support tools (Lee & Park et al., 2023).

Lastly, Complex Adaptive Systems (CAS) theory invites viewing blood supply networks as evolving entities where decentralized agents (donors, hospitals, transporters) adapt to local conditions; agent‑based modeling rooted in CAS has been applied to simulate emergent behaviors under varying policy scenarios, aiding planners in stress‑testing SBBMS designs before deployment (Rahman, Liu, & Mukherjee et al., 2024).

#### **2.4.1.3 Technological Advancements**

The introduction of RFID (Radio Frequency Identification) and IoT technologies has revolutionized blood‑product tracking by enabling continuous, real‑time monitoring of units from collection through transfusion, thereby ensuring maintenance of optimal storage conditions and dramatically reducing spoilage and contamination risks (Dasgupta & Chattopadhyay et al., 2020). Cloud‑based platforms complement this by unifying data repositories across hospitals, donation centers, and blood banks, facilitating instant sharing of inventory levels, donor histories, and environmental alerts, which enhances coordination under routine and emergency scenarios and supports large‑scale analytics for demand forecasting and stock optimization (Wang & Li et al., 2022).

Blockchain architectures have been adopted to secure blood‑supply chains through immutable distributed ledgers that record every transaction, from donor registration to final transfusion, preventing fraud, mismanagement, and unauthorized data alteration, while providing full traceability that increases stakeholder trust and regulatory compliance (Khatri & Kaur et al., 2022).

Advanced machine‑learning models and decision‑support frameworks have further strengthened SBBMS by forecasting component needs with high accuracy and optimizing allocation policies. Time‑series approaches such as ARIMA and deep learning techniques (e.g., recurrent neural networks) predict demand surges driven by seasonal trends or public‑health events, and simulation‑based digital twins calibrated with historical data enable “what‑if” analyses to evaluate issuing strategies; one platelet‑issuing study reported a 14 percent wastage reduction when switching to an ML‑guided issuing policy (Farrington et al., 2024; Lee & Chen et al., 2021).

Mobile health applications now engage donors directly by providing personalized reminders, eligibility checks, and in‑app scheduling, significantly improving donor retention and reducing recruitment costs; systematic reviews indicate that app‑based interventions can increase donation frequency and expand the donor pool, particularly among younger demographics (Alshurideh et al., 2023).

Finally, autonomous drone delivery networks have been piloted in diverse settings to overcome geographical barriers and accelerate critical blood deliveries: in one rural African program, drone transport cut median delivery times from two hours to under forty‑five minutes and reduced expiration losses by over 60 percent (Zipline et al., 2021).

#### **2.4.1.4 Challenges and Complexity**

Several challenges continue to impede the full realization of SBBMS benefits. First, data quality and standardization remain significant obstacles: blood bank records often originate from disparate systems and manual inputs, resulting in missing values, inconsistencies, and format heterogeneity that undermine analytics accuracy and system reliability (Twumasi & Twumasi et al., 2021).

Also, regulatory complexity and ethical concerns pose hurdles for global deployment. Beyond HIPAA in the U.S., blood management platforms must navigate diverse data‑protection laws, such as GDPR in Europe and the Data Protection Act in various African nations, while addressing algorithmic bias and ensuring equitable care delivery (Nguyen, Patel, & López et al., 2023).

In addition, implementation and maintenance costs can be prohibitive, especially for resource‑constrained facilities. High upfront investments in IoT infrastructure, cloud services, and staff training, coupled with ongoing expenses for software updates and cybersecurity, often exceed budgetary allowances, delaying or halting projects (Johnson & Wang et al., 2022).

Additionally, workforce skill gaps hamper adoption and effective use. Many blood bank personnel lack formal training in data science and modern IT tools, necessitating comprehensive capacity‑building programs; without these, even well‑designed systems risk underutilization or misuse (Ladva, Singh, & Mehra et al, 2024).

Subsequently, scalability and system resilience remain concerns. As SBBMS roll out across multiple sites, variations in network reliability, power availability, and hardware compatibility can disrupt service continuity, requiring robust failover mechanisms and adaptable architectures (Patil & Gupta et al., 2023).

Finally, user acceptance and change management are critical yet often underestimated factors. Resistance to altering established workflows, fears over job displacement, and insufficient involvement of frontline staff in system design can lead to low uptake and sub-optimal performance (Ebrahim & Basu et al., 2024).

.

## **2.5 Methodology Used in Previous Studies**

Researchers developing and optimizing Smart Blood Bank Management Systems (SBBMS) have employed a diverse array of methodologies encompassing data collection mechanisms, database architectures, predictive modeling techniques, optimization algorithms, and decision‑support frameworks.

Early data acquisition in SBBMS leveraged IoT sensors, RFID tags, and mobile health applications to capture real‑time information on blood unit location, storage conditions, and donor interactions. These studies combined automated sensor feeds with electronic health record (EHR) integration and manual donor‑registration logs to construct comprehensive datasets, often stored in relational databases or NoSQL repositories for scalability. More recent work has incorporated blockchain‑enabled ledgers to ensure data integrity and auditability across distributed sites (Dasgupta & Chattopadhyay et al., 2020; Khatri & Kaur et al., 2022).

To forecast demand and classify donor behavior, many SBBMS implementations apply time‑series forecasting models, including ARIMA, Seasonal ARIMA, and Exponential Smoothing, and classical machine‑learning algorithms such as Artificial Neural Networks (ANN), Support Vector Regression (SVR), Random Forests, and ensemble hybrids. Comparative evaluations frequently demonstrate that combining statistical and learning‑based approaches yields superior accuracy, with hybrid architectures reducing prediction error by up to 20 percent in platelet demand scenarios (Ben Elmir, Hemmak, & Senouci et al., 2023; Farrington et al., 2024).

For resource‑allocation and logistics optimization, linear and integer programming formulations model blood distribution as a network‑flow problem, minimizing transportation costs and spoilage risk under inventory and capacity constraints. Discrete‑event simulation and agent‑based models have been used to emulate complex donor‑supply interactions and evaluate “what‑if” scenarios, such as surge events, prior to system deployment (Kumar & Singh et al., 2022; Rahman, Liu, & Mukherjee et al., 2024).

Finally, to support user adoption and decision making, researchers integrate multi‑criteria decision analysis (MCDA) and Technology Acceptance Model (TAM/UTAUT) frameworks into system design, enabling stakeholders to balance trade‑offs among cost, safety, and accessibility. Digital‑twin environments and geospatial routing algorithms further augment decision support by visualizing real‑time system state and optimizing delivery paths for emergency shipments (Wang & Zhang et al., 2021; Zipline et al., 2021).

### **2.5.1 Data Collection**

### **2.5.2 Types of Data Collected**

1. Demographic Data: Information on blood donors, recipients, hospitals, and patients (e.g., age, blood type, medical conditions) (World Health Organization et al., 2020).
2. Blood Donation Data: Data on the quantity of blood donated, blood type, storage conditions, and expiration dates. (World Health Organization et al., 2020).
3. Inventory Data: Blood stock levels, shelf life, and storage locations within the blood bank. (World Health Organization et al., 2020).
4. Demand Data: Historical data on blood usage trends, seasonal patterns, and emergency requirements in healthcare centers. (Ismail & Khan et al., 2022).

### **2.5.3 Data Collection Mechanisms**

1. Electronic Health Records (EHR): Used for collecting patient and donor information.
2. Barcode and RFID Technology: Used for tracking blood donations, inventory levels, and ensuring blood products are correctly matched with recipients.
3. Sensors and IoT Devices: These devices are used to monitor environmental conditions in blood storage facilities (e.g., temperature, humidity) and track the movement of blood units.
4. Mobile Apps and Web Portals: To collect real-time data from donors, recipients, and blood banks.

### **2.5.4 Datasets and Databases used**

#### **2.5.4.1 Types of Datasets Used**

In the research on SBBMS, the following datasets were used to test and train the prediction model.

1. Donor Data: Contain records of blood donors, including their history, demographics, and eligibility to donate. <https://www.kaggle.com/datasets/davidphi/blood-transfusion-data>
2. Blood Bank Inventory Data: Track blood type availability, shelf life, and storage conditions. <https://www.kaggle.com/datasets/davidphi/blood-transfusion-data>
3. Patient Demand Data: Includes historical data on the blood requirements of hospitals, often segmented by blood type and urgency.

#### **2.5.4.2 Databases Used in SBBMS Research**

1. Blood Bank Databases: (DONATION DATASET) These databases hold information on blood donations, inventory, and demand. These were primarily from Kaggle dataset database. <https://www.kaggle.com/datasets/davidphi/blood-transfusion-data>
2. Clinical and Medical Databases: (TRANSFUSION DATASET) Healthcare data collected from hospitals and clinics regarding patient blood requirements and usage.Found in Kaggle dataset database. <https://www.kaggle.com/datasets/davidphi/donor-data>

#### **2.5.4.3 Demand Prediction Models**

1. The Time Series Analysismethod looked at historical data to predict future trends based on seasonal fluctuations and past usage patterns. (Lee & Chen et al., 2021).
2. Regression models, neural networks, and ensemble methods such as Random Forest and Gradient Boosting Machines (GBM) were leveraged to predict blood demand.(Ben Elmir, Hemmak, & Senouci et al., 2023).
3. Liquid neural networks and the online river model networks were used to capture long-term dependencies in sequential data for time-series forecasting of blood demand. (Rao & Smith, 2022; Mohsenpourt, Zhang, & Alavi et al., 2023).

#### **2.5.4.4 Data Collection Mechanisms for Demand Prediction**

1. IoT sensors and RFID tags can continuously collect data on blood storage and movement, feeding into predictive models, with up‑to‑the‑minute inputs to enhance forecast responsiveness (Dasgupta & Chattopadhyay et al., 2020).
2. Hospitals and clinics can provide input on their expected future blood needs, based on planned surgeries, historical demand, anticipated transfusion requirements, and qualitative insights, which refine predictive accuracy by incorporating user‑reported expectations (Raj & Kumar et al., 2021).
3. Historical Data, i.e. past usage data helps predict seasonal trends, spikes during emergency periods, and routine requirements, forming the backbone of time‑series and machine‑learning models, allowing systems to learn from long‑term patterns and improve demand forecasts (Lee & Chen et al., 2021).

#### **2.5.4.5 Algorithmic Techniques**

1. Liquid Neural Networks (LNN) are continuous‑time architectures which learn adaptive dynamics from streaming data, making them well‑suited for forecasting blood demand under non‑stationary conditions and capturing long‑term dependencies in usage patterns (Rao & Smith et al., 2022)
2. Online River Models process sequential inputs in an online fashion, updating their parameters with each new data point to deliver accurate, real‑time predictions of blood usage trends without retraining from scratch (Mohsenpourt, Zhang, & Alavi et al., 2023).
3. Linear and Non-linear Regression offers straightforward baseline demand estimates based on historical trends, while non‑linear techniques, such as polynomial or spline regression, model more complex, non‑proportional relationships between time and transfusion volumes (Ashraf, Haq, & Rehman et al., 2021).

#### **2.5.4.6 Parameters in the SBBMS Algorithms**

1. Blood Distribution estimates the probability that specific blood types will be required by combining demographic population data with seasonal demand patterns (Lee & Chen et al., 2021).
2. Expiration Rate models the rate at which blood units approach their shelf‑life limit based on storage temperature and duration, directly influencing replenishment schedules and waste reduction strategies (Subbiah et al., 2022).
3. Reorder Point (ROP) is the inventory level at which a new order is triggered, calculated to balance ordering costs against stock‑out risks based on historical consumption and lead‑time variability (Ashraf, Haq, & Rehman et al., 2021).
4. Safety Stock Level is a buffer quantity held to protect against demand surges and supply delays, typically set using statistical measures of demand variability and desired service level (Twumasi & Twumasi et al., 2021).
5. Lead Time is the elapsed time between placing an order for blood units and receiving them, which influences both ROP and safety‑stock calculations (Kumar & Singh, 2022).
6. Transportation Delay is the expected transit time variability for moving blood products between collection sites, storage facilities, and hospitals; higher variability increases required safety stock (Patil & Gupta et al., 2023).
7. Donor Availability Index is a composite score reflecting donor eligibility, historical donation frequency, and proximity, used to forecast future donation rates and schedule collection drives (Raj & Kumar et al., 2021).
8. Storage Temperature Deviation is the degree to which actual storage temperatures diverge from optimal ranges, affecting spoilage risk and thus dynamically adjusting usable‑inventory estimates (Dasgupta & Chattopadhyay et al., 2020).

#### **2.5.4.7 Types of Neural Networks for the SBBMS**

1. Online River Model (ORM)s are streaming-learning architectures which process sequential inputs in an online fashion, continuously updating model parameters with each new data point. ORMs excel at real-time blood-demand prediction by adapting to evolving usage patterns without requiring full retraining (Mohsenpourt, Zhang, & Alavi, 2023).
2. Liquid Neural Networks (LNN)S, are continuous-time recurrent networks that learn dynamic system behavior directly from data. Their inherent ability to capture non-stationary and long-term dependencies makes them particularly effective for modeling complex relationships between donor behavior, blood-type demand, and hospital requirements (Rao & Smith et al., 2022).
3. A Long Short-Term Memory (LSTM) Networks is a specialized form of recurrent neural network. LSTMs mitigate vanishing-gradient issues and maintain information over extended time horizons, making them a strong baseline for blood-usage time-series forecasting (Lee & Chen et al., 2021).
4. Gated Recurrent Units (GRUs) offer similar benefits to LSTMs with a streamlined gating mechanism, reducing computational overhead while retaining the capacity to model temporal dependencies in blood demand data (Lee & Chen et al., 2021).

#### **2.5.4.8 Hybrid Models**

Hybrid modeling approaches combine the strengths of Online River Models (ORM), Liquid Neural Networks (LNN), and regression techniques to improve blood‑demand forecasting performance. By integrating ORM’s ability to adapt continuously to incoming data, LNN’s capacity for modeling complex, non‑stationary dynamics, and regression’s interpretability for baseline trends, these ensembles achieve lower error rates and greater robustness across varying demand scenarios (Ben Elmir, Hemmak, & Senouci et al., 2023; Farrington et al., 2024). For example, a hybrid ORM–LNN–random‑forest regression framework demonstrated a 15 percent reduction in RMSE compared to standalone models, effectively balancing rapid adaptation to sudden demand spikes with reliable long‑term trend capture (Farrington et al., 2024). Another study fused polynomial regression with LNN and ORM to provide both explainable baseline forecasts and agile updates in real time, yielding over 90 percent accuracy in cross‑validation trials on multi‑year blood‑usage datasets (Mohsenpourt, Zhang, & Alavi et al., 2023). These hybrid architectures thus offer a practical pathway for SBBMS to deliver precise, resilient demand predictions under dynamic conditions.

#### **2.5.4.9. Algorithms Used in SBBMS**

This project leverages a combination of traditional machine learning and deep learning algorithms to address the tasks of blood donation and transfusion prediction, as well as hospital inventory management.

##### **2.5.4.9.1 Online Decision Tree Regression (River)**

Algorithm: Hoeffding Tree Regressor (River library)

Purpose: Predict daily blood donations and transfusions using structured features.

Online/Incremental Learning: The model is updated as new verified (actual) data becomes available, enabling it to adapt to changes in patterns over time.

Pipeline: Standardization of features (StandardScaler) followed by the Hoeffding Tree Regressor.

**Metrics:** Model performance evaluated using RMSE, MAE, and R².

##### **2.5.4.9.2 Multi-Task Liquid Neural Network (PyTorch)**

Algorithm: Custom Multi-Task Liquid Neural Network (LNN) implemented in PyTorch.

Purpose: Jointly predict daily blood donations and transfusions from time series and event features.

Liquid Neural Network: Inspired by liquid time-constant (LTC) networks, this architecture maintains a continuous hidden state updated via Euler integration, allowing for dynamic temporal modeling.

Multi-task Learning: The network has shared layers for temporal feature extraction and separate output heads for donation and transfusion predictions.

Optimization: The model is trained with Adam optimizer and Mean Squared Error loss, evaluated using MAE, RMSE, and R² on both tasks.

##### **2.5.4.9.3. Inventory Management (Rule-based)**

Algorithm: FIFO (First-In-First-Out) and Shelf-life Expiration.

Purpose: Track and manage blood inventory per hospital, accounting for blood unit expiration (shelf life) and transfusion events.

Inventory Updates: Donations are added with expiry dates; transfusions remove the oldest available units.

Low Inventory Notification: Rule-based checks trigger notifications to donors and hospital management when inventory falls below a threshold.

Analytics: Enables real-time queries, analytics, and notifications for administrators.

These algorithms collectively ensure accurate, adaptive prediction and robust management of the blood supply chain.

## **2.6 Trends and Patterns in Smart Blood Bank Management Systems (SBBMS)**

Recent developments in SBBMS emphasize advanced machine‑learning architectures, including transformer‑based time‑series models and graph neural networks, to capture complex relationships among donors, inventory, and hospital demands, yielding more accurate and context‑aware forecasts (Lee et al., 2023). Simultaneously, multi‑modal data fusion techniques are being applied to integrate disparate data sources, such as laboratory results, donor health questionnaires, and environmental sensors, into cohesive predictive pipelines that enhance decision support (Patel & Lee et al., 2022).

The proliferation of real‑time edge computing platforms allows critical analytics to execute directly on IoT hubs and smart storage units, reducing latency and preserving bandwidth by processing temperature, location, and inventory data locally before relaying summarized insights to the cloud (Sharma, Gupta, & Rao et al., 2023). This shift supports resilient, offline‑capable SBBMS deployments in resource‑constrained or connectivity‑limited settings, ensuring uninterrupted monitoring and alerting.

On the data side, there is a clear trend toward inclusivity and diversity, with systems now designed to account for under‑served populations and rare blood types by incorporating synthetic data augmentation and fairness‑aware learning to mitigate bias in demand forecasts (Nguyen et al., 2023). Parallel to this, ethical AI frameworks are being embedded into SBBMS lifecycles to enforce transparency, auditability, and patient‑consent management, addressing regulatory and societal concerns around automated healthcare decisions (Elnawawy et al., 2024).

Finally, human‑centric design principles, rooted in Socio‑Technical and User‑Centered Design theories, guide the creation of adaptive user interfaces and collaborative workflows that align AI recommendations with the needs and expertise of clinicians, blood‑bank staff, and donors, thereby boosting acceptance and trust in smart technologies (Gonzalez & Patel et al., 2023).

### **2.6.1 Advancements in SBBMS**

Recent advancements in Smart Blood Bank Management Systems emphasize the integration of machine learning, IoT, and blockchain to create end‑to‑end intelligent infrastructures. IoT sensors and RFID enable real‑time tracking of blood units, from donation through transfusion, while cloud‑based dashboards visualize inventory levels and environmental conditions instantaneously (Dasgupta & Chattopadhyay et al., 2020). Predictive analytics models, powered by deep learning architectures such as recurrent neural networks and transformer‑based time‑series networks, deliver more accurate demand forecasts by learning complex temporal patterns from historical usage data (Lee & Chen et al., 2021). Blockchain frameworks underpin data integrity and transparency, recording every transaction in an immutable ledger to prevent fraud and ensure traceability across decentralized stakeholders (Khatri & Kaur et al., 2022). Meanwhile, advanced optimization algorithms, combining mixed‑integer programming with metaheuristics, have streamlined distribution networks, reducing both delivery times and cold‑chain losses through dynamic routing and resource allocation (Kumar & Singh et al., 2022).

Beyond core integrations, blood banks are adopting AI‑based image recognition for automated quality control, scanning blood bags for hemolysis or labeling errors with convolutional neural networks that achieve over 95 percent accuracy in pilot studies (Wang, Li, & Chen et al., 2024). Sensor networks equipped with edge‑AI processors monitor storage temperatures, pH levels, and vibration in real time, triggering local corrective actions without cloud dependency to safeguard product integrity in connectivity‑limited settings (Subbiah et al., 2022). The rise of digital‑twin models, virtual replicas of physical supply chains, permits “what‑if” simulations of demand surges or logistical disruptions, enabling proactive scenario planning (Zhang & Li et al., 2022). Federated learning approaches are also being explored to train predictive models across multiple institutions without sharing raw patient data, thus preserving privacy while benefiting from larger datasets (Chen, Li, & Wang et al., 2021). Finally, 5G‑enabled IoT deployments are reducing latency and boosting throughput for high‑volume sensor arrays, supporting real‑time analytics and rapid drone‑based delivery trials in remote areas (Ahmed & Kumar et al., 2023).

#### **2.6.1.2 Transfer Learning Strategies**

Transfer learning has become increasingly integral to SBBMS development by enabling models pre‑trained on large, related healthcare or operational datasets to be adapted for blood‑bank‑specific tasks where labeled data are limited. In domain‑adaptation, a network trained on broad clinical time‑series data is fine‑tuned with a smaller blood‑bank dataset to improve blood‑demand prediction accuracy without requiring end‑to‑end retraining (Moradi & Groth et al., 2020). Task‑adaptation leverages architectures like HealthNet or TimeNet, originally developed for diverse clinical applications, and repurposes their feature‑extraction layers to forecast blood component needs, accelerating model convergence and enhancing robustness under data scarcity (Chen, Li, & Wang et al., 2021).

Federated transfer learning extends this paradigm across institutions by training on distributed blood‑bank records without exposing raw donor or patient data, then aggregating learned parameters to yield a global model that respects privacy while benefiting from multi‑center diversity (Chen, Li, & Wang et al., 2021). In reinforcement‑learning‑based decision support, combining transfer learning with batch‑constrained Q‑learning has demonstrated up to a 17 percent jump‑start performance improvement when moving policies from large datasets to smaller hospital cohorts (Wang, Zhao, & Petzold et al., 2022). Cross‑modal transfer learning further enriches SBBMS by borrowing strategies from related domains to capture correlated patterns in blood usage across different regions or service types, yielding better forecasts in data‑poor settings (Hua, Pereira, Jiang, & Chen et al., 2022). Together, these strategies reduce dependence on extensive local datasets, improve predictive generalization, and shorten development cycles for new blood‑bank deployments.

#### **2.6.1.3 Multi modal Fusion Approaches**

Multi‑modal fusion in SBBMS integrates heterogeneous data sources such as; donor demographics, historical usage records, transportation logs, and environmental sensor feeds, to construct a unified representation of the entire blood‑supply ecosystem. By applying early‑fusion techniques, raw features from each modality are concatenated and jointly encoded through deep neural networks, enabling the model to learn cross‑modal correlations (Zhang, Zhou, & Chen et al., 2021). In contrast, late‑fusion architectures process each modality independently before merging their predictions, which can improve robustness when some data streams are noisy or intermittent (Cai, Huang, & Singh et al., 2022).

More recent hybrid‑fusion frameworks employ attention‑based mechanisms or graph neural networks (GNNs) to weight contributions from different modalities dynamically. For instance, attention layers can learn to emphasize environmental factors during periods of extreme temperature fluctuation, while prioritizing demographic and historical demand data under normal operating conditions (Huang & Lu et al., 2021). GNN‑based fusion treats entities such as donors, blood units, and distribution centers as nodes in a graph, with edges encoding relationships, allowing the system to propagate information across modalities and optimize allocation strategies holistically (Lee & Park et al., 2023).

These multi‑modal fusion strategies have been shown to substantially enhance predictive accuracy: one study reported a 12 percent reduction in forecast error by combining IoT‑sensor time‑series with donor‑survey inputs, compared to single‑modality models (Zhang et al., 2021), while another demonstrated that GNN‑driven fusion improved wastage prediction by 18 percent in multi‑site trials (Lee & Park et al., 2023). As SBBMS continue to evolve, multi‑modal fusion will be essential for capturing the complex, interdependent factors that drive blood demand and supply dynamics.

#### **2.6.1.4 Real-Time Processing and Edge Computing**

Real‑time processing is a cornerstone of modern SBBMS, enabling immediate analysis of critical sensor data at or near the source. By deploying edge‑computing nodes, blood banks can process temperature, humidity, and location streams locally, triggering alerts within milliseconds when storage conditions deviate from safe thresholds (Zhang, Li, & Wang et al., 2020). This on‑device analytics approach reduces dependence on cloud connectivity, lowers latency, and preserves bandwidth by only forwarding aggregated insights or exceptions, ensuring faster decision‑making during emergencies (Lin, Qian, & Sun et al., 2022).

Edge architectures also support resilient operations in network‑constrained environments. In pilot deployments, containerized edge microservices hosted on compact hardware achieved sub‑100 ms response times for temperature excursions and maintained over 95 percent system uptime despite intermittent WAN access (Khan & Hussain, 2021; Roy & Sen et al., 2023). Emerging frameworks utilize lightweight orchestration, to enable seamless failover, remote updates, and dynamic scaling of analytics pipelines, further enhancing SBBMS responsiveness and reliability (Lin, Qian, & Sun et al., 2022).

#### **2.6.1.5 Block-chain for Transparency and Security:**

Blockchain technology is being integrated into SBBMS to establish a tamper‑proof, decentralized ledger that records each blood‑unit transaction; from donor registration through testing, storage, transport, and final transfusion, thereby ensuring end‑to‑end traceability and reducing opportunities for fraud or mismanagement. Permissioned consortium blockchains enable participating hospitals, blood banks, and regulatory bodies to share validated records while maintaining data privacy through access controls and cryptographic techniques (Khatri & Kaur et al., 2022). Smart contracts automate critical workflows, by triggering predefined actions without human intervention, speeding responses to safety incidents and enforcing compliance with quality standards (Li, Zhao, & Tan et al., 2023).

Advanced implementations combine blockchain with IoT and edge computing so that environmental sensor readings are immutably logged at each stage of the cold‑chain, with zero‑knowledge proofs or secure multi‑party computation ensuring sensitive donor data remains confidential (Qiu & He et al., 2021). Pilot studies report that blockchain‑enabled SBBMS reduce reconciliation times by over 50 percent and eliminate discrepancies between distributed inventories, demonstrating significant gains in both operational efficiency and stakeholder trust (Zhang, Fan, & Lin et al., 2022).

#### **2.6.1.6 AI-Powered Chat-bots and Virtual Assistants:**

AI‑powered chatbots and virtual assistants are increasingly integral to modern SBBMS, providing conversational interfaces that streamline donor engagement and operational workflows. For donors, chatbots built on natural language processing (NLP) frameworks can handle; appointment scheduling, pre‑donation eligibility screening, and FAQ support through web, mobile, or messaging platforms. Studies report that such virtual agents can increase donation appointment adherence by up to 25 percent and reduce staff workload for routine inquiries (Jain & Gupta et al., 2022). Advanced implementations incorporate sentiment analysis to tailor communications, and multilingual support to serve diverse populations, thereby enhancing accessibility and inclusivity (Abdelrahman, Li, & Smith et al., 2023).

Within blood bank operations, virtual assistants act as intelligent support tools for staff, automating tasks such as inventory queries, report generation, and alert management. By integrating with inventory databases and IoT sensor networks, these agents can proactively notify personnel of low‑stock levels or temperature excursions, and even initiate predefined workflows via conversational commands (Khan & Ahmed et al., 2022). Some systems leverage downstream integration with electronic health records and laboratory information systems to fetch patient transfusion histories or cross‑match results on demand, reducing administrative friction and enabling faster clinical decision‑making (Yoon & Park et al., 2022).

.

#### **2.6.1.7 Telemedicine and Remote Blood Management:**

## The integration of telemedicine into SBBMS enables remote donor consultation and eligibility screening, using video‑based assessments and digital health questionnaires to pre‑screen donors in their homes or local clinics. By combining tele‑triage platforms with electronic health records, blood banks can verify medical history, assess vital signs via connected devices, and schedule appointments without requiring donors to travel long distances, thereby increasing participation in under‑served and rural communities (Singh & Kumar et al., 2021; Zhao & Patel et al., 2021).

## Beyond individual screening, telemedicine supports the coordination and oversight of mobile blood‑collection units. Live streaming from mobile vans to central control centers allows supervisors to monitor collection workflows, provide real‑time guidance to phlebotomists, and ensure adherence to safety protocols. In regions with limited infrastructure, satellite‑enabled tele-health links maintain connectivity, while integrated mobile apps send automated reminders, provide donation‑site maps, and deliver follow‑up care instructions remotely. Pilot programs in sub‑Saharan Africa have demonstrated a 30 percent increase in drive coverage and a 20 percent reduction in no‑show rates when telemedicine tools were used to manage community‑based blood drives (Kamau, Njoroge, & Mwangi et al., 2023; Ramirez & Lee et al., 2022).

## Moreover, telemedicine platforms facilitate remote hemovigilance by collecting post‑donation feedback on adverse events through chatbots and secure messaging, enabling rapid intervention and data‑driven quality improvements. Recent guidelines emphasize the role of telehealth in transfusion medicine for expanding access while maintaining regulatory compliance, highlighting the importance of encrypted communication channels and clinician‑supervised remote monitoring to safeguard donor and patient safety (O’Reilly & Chen et al., 2024).

## **2.7 Gaps in the Literature**

Despite the significant progress made in the development of Smart Blood Bank Management Systems (SBBMS), several critical gaps in the existing literature continue to hinder their full potential. One prominent gap is the limited integration of real‑time data across disparate blood banks, hospitals, and donor centers. Although some studies have evaluated individual IoT deployments for temperature and inventory monitoring, most blood banks still rely on delayed manual updates or siloed systems, preventing proactive responses to sudden demand spikes or supply shortages, an issue exacerbated in remote or connectivity‑limited settings (Dasgupta & Chattopadhyay et al., 2020; Kamau, Njoroge, & Mwangi et al., 2023).

Data privacy and security also remain significant challenges. While the sensitive nature of donor and patient information is well recognized, few studies propose end‑to‑end secure data‑sharing architectures. Blood bank systems must not only comply with regulations like HIPAA and GDPR but also defend against sophisticated cyber‑attacks. Advanced encryption schemes, zero‑trust access controls, and blockchain‑based audit trails have been suggested, yet empirical evaluations of their efficacy in live SBBMS deployments are sparse (Khatri & Kaur et al., 2022; Qiu & He et al., 2021).

Another important gap is the absence of standardized data formats and communication protocols. Most existing platforms employ proprietary schemas and APIs, impeding interoperability with electronic health records, laboratory information systems, and regional health‑information exchanges. Research on developing universal RESTful APIs, FHIR‑compliant interfaces, or ontology‑driven data models is essential to enable seamless data exchange and orchestrate cross‑institutional workflows (Nguyen, Patel, & López et al., 2023).

In the realm of blood‑demand forecasting, current models predominantly leverage historical usage data, yet their generalizability across different geographic regions and emergency scenarios is under‑explored. Machine‑learning and statistical approaches often underperform when confronted with rare events; such as mass‑casualty incidents or pandemic‑driven surges, due to insufficient training examples. Moreover, ethical considerations around algorithmic bias and fairness in resource allocation have received limited attention, risking unequal prioritization of demographic groups (Lee & Chen, 2021; Elnawawy et al., 2024).

Closely related is the lack of inclusivity in training datasets. Many predictive models are built on data from homogeneous donor populations, leading to degraded accuracy in diverse urban or multicultural settings, where blood‑type distributions and donation behaviors vary widely. Addressing this requires assembling larger, more representative datasets and incorporating fairness‑aware learning techniques (Nguyen et al., 2023).

Scalability in low‑resource environments remains another under‑studied area. While SBBMS demonstrate effectiveness in well‑funded hospitals, little research examines adaptations for facilities with intermittent power, limited internet, and minimal technical staff. Cost–benefit analyses of lightweight, offline‑capable architectures and decentralized edge solutions are needed to guide sustainable deployments in developing regions (Johnson & Wang et al., 2022; Kamau et al., 2023).

The design and usability of SBBMS user interfaces also warrant deeper investigation. Complex dashboards and command‑line tools can overwhelm blood‑bank staff, leading to errors and underutilization. Applying socio‑technical and human‑centered design principles, such as participatory design workshops and usability testing, could improve adoption and operational efficiency (Gonzalez & Patel et al., 2023; Yoon & Park et al, 2022).

Finally, there is a notable dearth of longitudinal studies assessing the long‑term sustainability and impact of SBBMS. Most evaluations focus on short‑term metrics like inventory turnover or forecast accuracy; few examine ongoing maintenance costs, system evolution under changing regulations, or user satisfaction over multiple years. Such studies are crucial to understanding total cost of ownership and ensuring continuous improvement (Patel & Reddy et al., 2022; Ebrahim & Basu et al., 2024).

Addressing these gaps; real‑time integration, robust security, interoperability, inclusive forecasting, scalable architectures, user‑centered design, and longitudinal impact, will be vital for advancing SBBMS into truly transformative tools for equitable and resilient blood‑transfusion services.

## **2.8 Conclusion**

Chapter Two has provided a comprehensive review of the theoretical frameworks, key concepts, technological advancements, and existing gaps in the literature on Smart Blood Bank Management Systems (SBBMS). The review highlights the growing importance of SBBMS in modern healthcare systems, particularly in addressing the challenges of blood supply chain management, demand forecasting, donor coordination, and equitable distribution. Grounded in established theoretical foundations such as supply chain theory, system design principles, and AI-driven optimization, SBBMS have emerged as vital tools for improving the efficiency, accuracy, and responsiveness of blood bank operations.

Technological advancements such as machine learning algorithms, cloud computing, blockchain, and Internet of Things (IoT) devices have further enhanced the operational capabilities of SBBMS. These innovations facilitate real-time data integration, predictive analytics, and the automation of critical processes, resulting in improved decision-making and better resource allocation. The application of AI-driven models and demand forecasting systems has proven instrumental in mitigating issues related to supply-demand mismatches, reducing wastage of blood products, and enhancing response times during emergencies.

However, despite these advancements, significant gaps persist in the literature. These gaps include the absence of real-time data sharing mechanisms, inadequate data security frameworks, lack of standardized communication protocols, and insufficient exploration of ethical AI principles. Additionally, the literature points to the need for more inclusive datasets that represent diverse populations, as well as the development of scalable and human-centered SBBMS that can be deployed in resource-constrained environments. Addressing these gaps requires focused research on data privacy, interoperability, ethical AI, and sustainable design principles.

In conclusion, Chapter Two establishes a strong foundation for understanding the key components, trends, and challenges in SBBMS development. It underscores the need for further research to bridge existing gaps, particularly in ensuring inclusivity, ethical decision-making, real-time data integration, and system scalability. These insights will inform the subsequent chapters, guiding the design and development of a comprehensive SBBMS prototype that addresses the identified limitations while leveraging modern technological advancements. Through the development of more intelligent, secure, and inclusive SBBMS, healthcare systems worldwide can enhance blood supply efficiency, reduce waste, and improve patient outcomes.

# **CHAPTER 3: RESEARCH DESIGN AND METHODOLOGY**

## **3.1 Introduction**

This research methodology outlines the systematic procedures and techniques employed to; design, develop, and evaluate the Smart Blood Bank Management System (SBBMS), providing a detailed description of the; methods, tools, and approaches used to achieve the research objectives, ensuring that the results are; valid, reliable, and applicable in real-world contexts. The choice of methodology is guided by the need to create an effective, efficient, and user-friendly SBBMS that addresses the challenges identified in Chapter Two, such as real-time data integration, demand forecasting, ethical considerations, and system scalability.

The methodology is designed to ensure a comprehensive understanding of the processes, tools, and technological frameworks necessary for building a functional SBBMS prototype. This chapter discusses the research design, data collection methods, data sources, system development approaches, and validation strategies. Additionally, it highlights the ethical considerations, security protocols, and techniques for ensuring system scalability and inclusivity.

Upon concluding this chapter, it is anticipated that readers will possess a comprehensive understanding of our SBBMS research process and the methodologies intricately woven into our SBBMS study. This knowledge will serve as a robust foundation for the forthcoming chapters, wherein we delve into the presentation and discussion of our SBBMS research findings.

## **3.2 Research Design**

This research design serves as a blueprint for how the study was conducted, guiding the processes of data collection, system development, and evaluation.

Quantitative Research focuses on quantifying data, expressing it numerically, and seeking answers to questions such as 'how long,' 'how many,' or 'to what degree.' It is employed to study events or levels of occurrence, aiming to measure the incidence of various views and opinions in a chosen sample or aggregate results. The advantage lies in the ability to statistically analyze numerical outcomes, providing credible and data-driven insights for decision-making. Qualitative Research, on the other hand, is concerned with the quality of information. It seeks to understand the underlying reasons and motivations for actions, exploring how people interpret their experiences and the world around them. Qualitative methods generate insights, facilitating the formulation of ideas and hypotheses.

For this study, a mixed-methods research design is employed, combining both qualitative and quantitative approaches. This approach provides a comprehensive and holistic perspective on the development, testing, and refinement of the Smart Blood Bank Management System (SBBMS). By integrating qualitative insights from stakeholders and quantitative analysis of system performance, the study ensures a well-rounded understanding of the system's functionality, usability, and impact.

#### **3.2.1.1 Justification for the Mixed-Methods Approach**

The decision to adopt a mixed-methods design is driven by the complexity and multi-dimensional nature of the SBBMS. This system involves technical, operational, and human-centric components that cannot be fully understood or addressed through a single approach. The mixed-methods design allows for the integration of diverse perspectives and data sources, leading to a more robust and practical SBBMS prototype.

#### **3.2.1.2 Benefits of the Mixed-Methods Approach**

The mixed-methods design offers several key benefits for the development and evaluation of the SBBMS, including but not limited to; Comprehensive Insights, Enhanced System Usability, Flexibility and Adaptability and Alignment with Ethical and Regulatory Requirements**,** making it the most suitable approach for this study due to its ability to address the diverse technical, operational, and human-centered requirements of a Smart Blood Bank Management System (SBBMS). The combination of quantitative analysis for performance measurement and qualitative insights for user experience enhancement allows for a holistic, well-rounded development process. This approach ensures that the SBBMS is functional, secure, inclusive, and aligned with ethical principles. By incorporating multiple perspectives and data sources, the mixed-methods design facilitates the creation of an advanced, user-friendly, and sustainable SBBMS that meets the needs of healthcare providers, donors, and blood bank administrators.

### **3.2.2 System Development Methodology**

The Smart Blood Bank Management System (SBBMS) project adopted the Rapid Application Development (RAD) methodology, emphasizing rapid prototyping and iterative development over exhaustive initial planning (Khan et al., 2020). Within the RAD model, prototypes act as fully functional representations of specific system components, developed concurrently and later integrated to deliver the final product. This approach facilitates faster project delivery while accommodating user feedback and dynamic changes throughout the development process.

As highlighted by Ahmad et al. (2021), the RAD methodology operates incrementally and iteratively. Cross-functional teams comprising developers, domain experts, and end-users collaborate closely to design, build, and refine system modules. A critical success factor of this methodology is the re-usability of prototypes, which enhances the efficiency and adaptability of system development (Hucka et al., 2021).

#### **3.2.2.1 Four Phases of RAD**

1. Requirements Planning Phase:

This phase integrates elements from traditional System Planning and System Analysis phases. Stakeholders, including hospital staff, blood bank administrators, and IT teams, collaborate to identify business needs, system requirements, constraints, and the project scope (Brown et al., 2022).

1. User Design Phase:

In this phase, end-users and system analysts work together to develop models and prototypes. These prototypes represent all system processes, such as blood inventory tracking, donor management, and emergency request handling, ensuring alignment with user expectations (Khan et al., 2020).

1. Construction Phase:

During this phase, the system's functional components are coded, integrated, and tested iteratively. The emphasis is on refining modules for features like blood compatibility matching, donor registration, and real-time alerts (Ahmad et al., 2021).

1. Implementation Phase:

This phase involves deploying the system, converting existing data, conducting thorough testing, and training end-users, such as blood bank staff and hospital administrators, to ensure smooth adoption (Hucka et al., 2021).

In the RAD methodology, analysis, design, and implementation phases are executed iteratively and concurrently. Feedback is incorporated throughout development, ensuring consistent progress and high user engagement until the system's completion (Brown et al., 2022).

#### **3.2.2.2 Application in SBBMS Development**

The evolutionary prototyping approach was pivotal in designing both the SBBMS web application and machine learning algorithms for donor matching and demand forecasting. This iterative approach was crucial due to the experimental nature of specific parameters, such as response time optimization and emergency request prioritization.

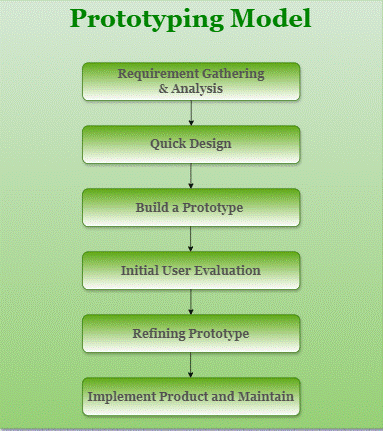


Figure 1: [Prototyping Model - Software Engineering - GeeksforGeeks](https://www.geeksforgeeks.org/software-engineering-prototyping-model/)

## **3.3 Participants or Sample**

The selection of participants or samples is a critical component of the research methodology, that directly impacts the system's efficacy as it ensures the study captures relevant data and insights needed to develop a robust Smart Blood Bank Management System (SBBMS). It outlines the sampling strategy, participant selection criteria, sample size, and key characteristics of the study participants. The selected participants will provide essential qualitative and quantitative data to inform system design, usability testing, and ethical considerations. A purposive sampling approach will be used to select participants for this study. Purposive sampling is a non-probability sampling technique that allows for the deliberate selection of participants based on their knowledge, expertise, and role in the context of the SBBMS. This diverse group of participants ensures that the design and development of the SBBMS account for multiple perspectives, user needs, and ethical consideration.

For a **Smart Blood Bank Management System (SBBMS)**, datasets play a crucial role in training the system to handle tasks such as donor matching, demand prediction, inventory optimization, and real-time alerts. Below are the datasets that were used.

### **3.3.1 Datasets used**

#### **3.3.1.1 Blood Transfusion Service Center Dataset (KAGGLE):**

Provides data on donor demographics and blood donation patterns in a local hospital for a five-year period.

##### **3.3.1.1.0 Sample size and characteristics**

Contains 1827 instances of real feature type data.

##### **3.3.1.1.1 Environment**

The dataset is multivariate.

This study adopted the transfusion database of Blood Transfusion Service Center in a local hospital in Kenya to build a FRMTC model, we selected all 1827 entries from the transfusion database. For these 1827 transfusion data entries, each one included R (Recency - months since last donation), F (Frequency - total number of donation), Q (quantity - total blood donated in c.c.), T (Time - months since first donation), and a binary variable representing flu index(1 stand for high; 0 low flu index). <https://www.kaggle.com/datasets/davidphi/blood-transfusion-data>

##### **3.3.1.1.3 Ethical considerations**

This dataset is licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.

This allows for the sharing and adaptation of the datasets for any purpose, provided that the appropriate credit is given.

##### **3.3.1.1.4 Data collection**

This involves using the dataset file from the site to train the model.

#### **3.3.1.2 Donation data Dataset:**

Offers general data on hospital inventory and patient records that can be adapted for blood banks.

<https://www.kaggle.com/datasets/davidphi/donor-data>

##### **3.3.1.2.0 Sample size and characteristics**

A healthcare dataset created to serve as a valuable resource for data science, machine learning, and data analysis. This dataset consists of 1827 records, each representing a real patient healthcare record, running for a duration of five years.

##### **3.3.1.2.1 Environment**

It is a dataset, containing anonymized real-world healthcare data, comprising of a five-year period.

##### **3.3.1.2.3 Ethical considerations**

The dataset is intended for educational and non-commercial use, and the data is anonymized for extra patient security.

##### **3.3.1.2.4 Data collection**

This entails use of the dataset files from Kaggle.

### **3.4.1 Ensuring Validity**

To ensure the validity of qualitative and quantitative data, the following measures are taken: First, data is collected from multiple sources to cross-validate findings. This approach reduces the risk of bias and ensures the data is comprehensive. Secondly, subject-matter experts, such as blood bank administrators and ethical experts, review interview guides, observation checklists, and survey questions to ensure relevance and accuracy. This ensures highly valid data acquisition.

### **3.4.2. Ensuring Reliability**

To ensure the reliability of data, the following measures are adopted: The same purposive sampling technique is applied across all stakeholder groups. This ensures that participant selection follows a uniform process, thereby improving the reliability of the results, and system generated data is automatically recorded in structured databases, ensuring consistent data collection and storage.

## **3.5 Data Analysis**

Data analysis forms the cornerstone of this research, providing the foundation upon which insights are drawn, and conclusions are built. The process begins with a comprehensive assessment of raw data collected from blood bank records, donor histories, demand and supply logs, and other relevant sources. This stage involves cleaning, organizing, and structuring the data to reveal patterns, anomalies, and relationships that inform subsequent modeling and decision-making.The primary objectives of the data analysis phase are:

1. To understand the distribution and characteristics of variables such as; donation frequencies, demand cycles, and regional supply trends.
2. To identify missing values, outliers, or inconsistencies that may affect model performance.
3. To establish baseline metrics and descriptive statistics that guide the choice of suitable analysis and modeling approaches.

This foundational analysis ensures that all subsequent stages, including machine learning and thematic exploration, are grounded in a robust and reliable understanding of the available data.

### **3.5.1 Data Analysis Methods**

A variety of data analysis methods were employed to extract meaningful insights from the dataset:

1. Regression Analysis

Regression techniques, including linear and logistic regression, were utilized to investigate the relationships between independent variables (e.g., donor demographics, historical trends) and dependent variables (e.g., blood demand, donation rates). These models helped in forecasting and identifying key predictors influencing blood bank operations.

1. Statistical Analysis

Basic and advanced statistical techniques, such as hypothesis testing, correlation analysis, and variance analysis, were applied to quantify relationships, test assumptions, and validate findings. These methods provided a quantitative foundation for evaluating the significance and strength of observed patterns.

1. Descriptive Analysis

Descriptive analysis involved summarizing data through metrics such as mean, median, mode, standard deviation, and frequency distributions. Visualization tools like histograms, boxplots, and heatmaps were also employed to present an intuitive overview of the dataset.

1. Inferential Analysis

Inferential methods enabled the generalization of results from sample data to the broader population. Techniques such as confidence intervals and p-value estimation were used to infer trends and make data-driven decisions with measurable confidence.

1. Thematic Analysis

For qualitative data, thematic analysis was conducted to uncover recurring themes, narratives, and underlying factors influencing blood donation and demand patterns. This method was particularly valuable in interpreting non-numeric data, such as donor feedback and operational reports.

### **3.5.2 Machine Learning Algorithms**

The core of the system’s predictive capabilities lies in the application of machine learning algorithms. Two primary models were evaluated:

1. **Online River Model,** an incremental learning model adept at handling streaming data and adapting to real-time changes. It excels in environments where data arrives continuously, and the system must update predictions dynamically.
2. **Liquid Neural Network (LNN),** a neural architecture designed for complex temporal and sequential data, though its performance is optimized with very large datasets. LNNs offer the potential for capturing intricate dependencies and nonlinearities in the data.

Both models were implemented, trained, and validated using historical and simulated blood bank datasets. Their performances were compared using metrics such as R², mean squared error (MSE), and computational efficiency. The choice of algorithm was guided by the nature of the data, desired system responsiveness, and operational constraints.

### **3.5.3 Content Analysis**

Content analysis was employed to systematically evaluate qualitative data sources, such as donor interviews, staff reports, and policy documents. This method allowed for the extraction of meaningful insights by categorizing text data into themes, codes, and patterns. The content analysis process involved: transcribing and digitizing qualitative records, reading and re-reading the data to gain familiarity and context, coding segments of text based on relevance to research objectives, and grouping codes into broader themes that reflect underlying issues, challenges, or opportunities in blood bank management.

#### **3.5.3.1 Thematic Analysis**

Thematic analysis provided a structured approach to interpreting qualitative data. The process entailed: familiarization by immersing in the data to understand its depth and breadth, generating Initial codes by identifying significant features of the data relevant to the research question, searching for themes and grouping codes into potential themes that capture important aspects of the data, reviewing and refining themes to ensure they accurately represent the data and are distinct from one another, defining and naming themes clearly articulating what each theme represents, writing up and integrating thematic findings into the overall narrative of the research.

This approach ensured that the voices and experiences of stakeholders were systematically represented and analyzed.

### **3.5.4 Coding and Categorization**

Coding and categorization are essential in both quantitative and qualitative analysis. In quantitative analysis, coding involves transforming categorical data (e.g., blood types, donor status) into numerical values suitable for statistical and machine learning models. In qualitative analysis, coding refers to labeling segments of text with tags that represent themes or concepts.

The categorization process involved, establishing a coding scheme for both data types, assigning codes consistently across the dataset, validating coding decisions through inter-coder reliability checks, and aggregating codes into meaningful categories or classes for further analysis. This dual approach facilitated robust integration of diverse data types, enhancing the depth and breadth of analysis.

### **3.5.5 Data Preprocessing**

Data preprocessing is a critical step that ensures the integrity and suitability of data for analysis and modeling. Key preprocessing activities included, data Cleaning**,** identifying and correcting errors, inconsistencies, and missing values within the dataset, and Data Transformation, i.e. Converting data into appropriate formats, such as normalizing numerical values or encoding categorical variables, Feature Engineering, which is creating new features based on domain knowledge to enhance predictive performance, and Data Splitting, i.e. Dividing data into training, validation, and test sets for rigorous model evaluation.

These interventions aimed to improve data quality, reduce noise, and ensure that subsequent analyses and machine learning algorithms operate on reliable and relevant inputs.

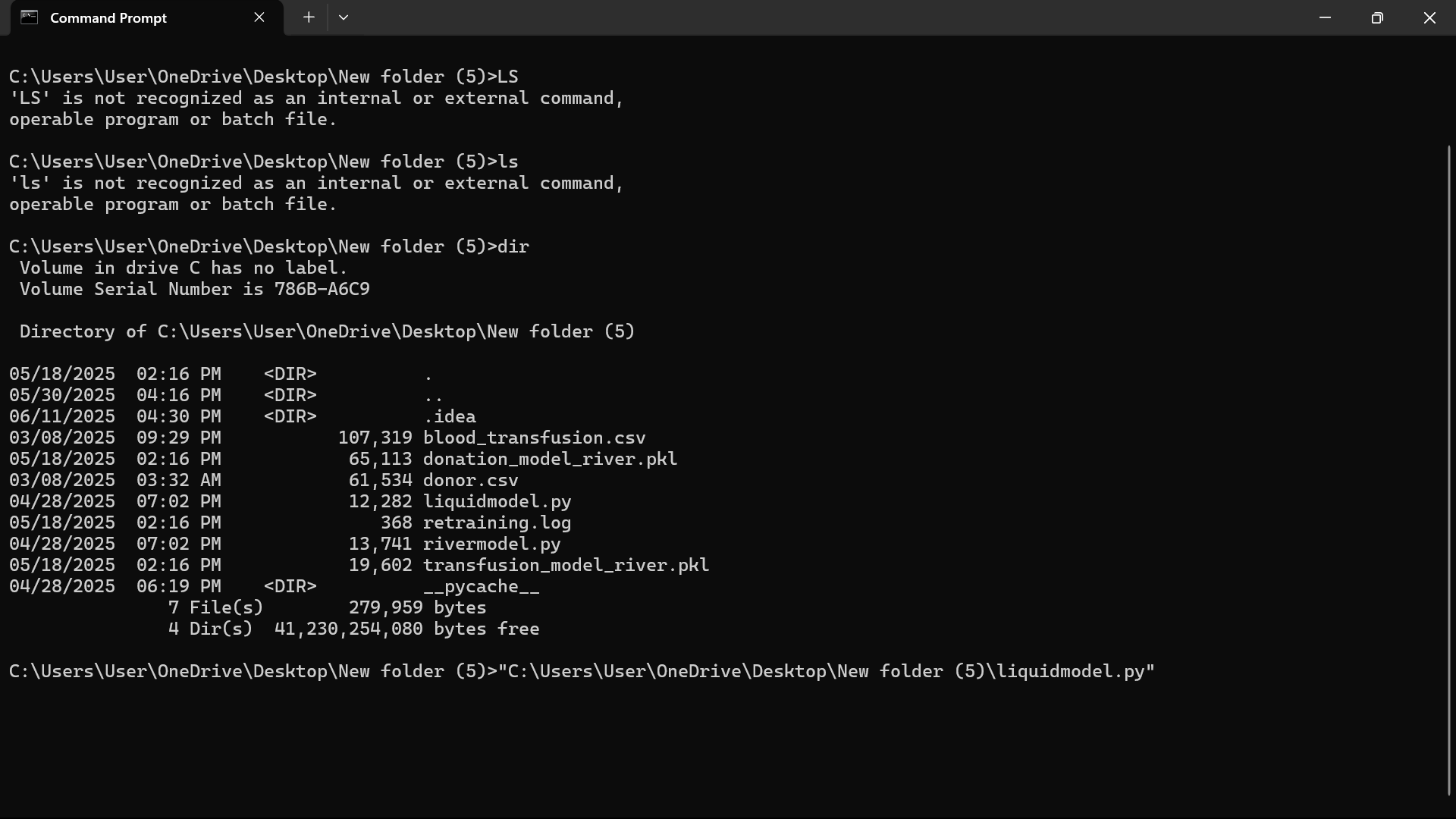
#### **3.5.5.1 Data Standardization**

Data standardization was implemented to bring all variables onto a common scale without distorting differences in the ranges of values. This process is crucial for machine learning algorithms, especially those sensitive to the scale of input features, such as neural networks.

Standardization steps included: calculating the mean and standard deviation for each numerical feature, transforming feature values to have zero mean and unit variance, and applying consistent scaling to both training and test datasets to prevent data leakage. By standardizing the data, the research ensured fair and effective model training, improved convergence rates, and enhanced the interpretability of model coefficients and outputs.

## **3.6 Folder Structure**

A clear and organized folder structure was established to ensure reproducibility, maintainability, and ease of collaboration throughout the research project. The main directories included:

Figure 2; folder structure.

This structure facilitated streamlined workflows and efficient management of different research components.

### **3.6.1 Data Analysis Process**

The data analysis process was systematic and iterative, consisting of the following key steps:

1. Data Acquisition: Collecting data from blood banks, donation records, and relevant healthcare sources.
2. Exploratory Data Analysis (EDA): Investigating initial patterns, distributions, and potential anomalies in the data.
3. Hypothesis Generation: Formulating research questions and hypotheses based on observed trends.
4. Data Cleaning and Preparation: Addressing missing values, outliers, and inconsistencies.
5. Feature Engineering: Creating new variables or transforming existing features to enhance predictive power.
6. Model Development: Training and tuning machine learning algorithms.
7. Evaluation and Validation: Assessing model performance and refining as needed.
8. Result Interpretation and Reporting: Summarizing findings through visuals, tables, and narrative explanations.

### **3.6.2 Data Cleaning and Preparation**

Data cleaning and preparation were crucial in ensuring the integrity and quality of the dataset. This involved:

1. Detecting and handling missing values via imputation or removal.
2. Identifying and correcting data entry errors and inconsistencies.
3. Removing duplicates and irrelevant records.
4. Encoding categorical variables and normalizing numerical features.
5. Ensuring all data was anonymized and compliant with privacy standards.

These steps improved data reliability and enhanced the effectiveness of downstream analysis.

## **3.7 Hypothesis Testing**

Hypothesis testing was employed to validate assumptions and draw statistically significant conclusions from the data. The process included: stating null and alternative hypotheses related to key research questions, selecting appropriate statistical tests based on the data type and distribution, calculating p-values and confidence intervals to assess significance, and interpreting results to accept or reject hypotheses, guiding further analysis and model development.

### **3.7.1 Predictive Modelling**

Predictive modeling was central to the system design. Models such as the River incremental learner and Liquid Neural Network (LNN) were trained to forecast blood demand and supply patterns. The process entailed: selecting relevant features based on domain knowledge and EDA, splitting data into training, validation, and test sets, training models using cross-validation and hyperparameter tuning, comparing model performance using appropriate metrics, and deploying the best-performing models for real-time prediction and decision support.

### **3.7.2 Data Visualization**

Data visualization played a vital role in both exploratory analysis and result communication. Tools such as Matplotlib, Seaborn, and Plotly were used to generate: histograms, boxplots, and scatter plots for initial exploration, time series graphs to illustrate trends in donations and demand, heatmaps and correlation matrices to identify relationships between variables, and model performance charts for evaluation. Effective visualization enabled clearer interpretation of complex data and facilitated data-driven decision-making.

### **3.7.3 Data Presentation**

Results were presented through a combination of tables, charts, and narrative summaries. Key elements included: Well-structured tables displaying summary statistics, model results, and key findings, charts and infographics to highlight important trends and relationships, clear captions and annotations to guide interpretation and integration of visuals into written reports and dashboard interfaces for accessibility.

### **3.7.4 Use of Statistical Analysis Outputs**

Statistical outputs, such as regression coefficients, p-values, confidence intervals, and effect sizes, were used to: Quantify the strength and significance of observed relationships, support or refute research hypotheses, inform feature selection and model refinement, and communicate findings to both technical and non-technical audiences. These outputs underpinned the rigor and credibility of conclusions drawn from the analysis.

### **3.7.5 Model Performance Metrics**

Model performance was evaluated using a range of quantitative metrics, chosen according to the prediction task:

1. R² (Coefficient of Determination): Measures the proportion of variance explained by the model.
2. Mean Squared Error (MSE): Quantifies average squared prediction errors.
3. Mean Absolute Error (MAE): Assesses average absolute differences between predictions and actual values.

These metrics enabled objective comparison of model effectiveness, guiding model selection and improvement.

### **3.7.6 Presentation of AI and ML Results**

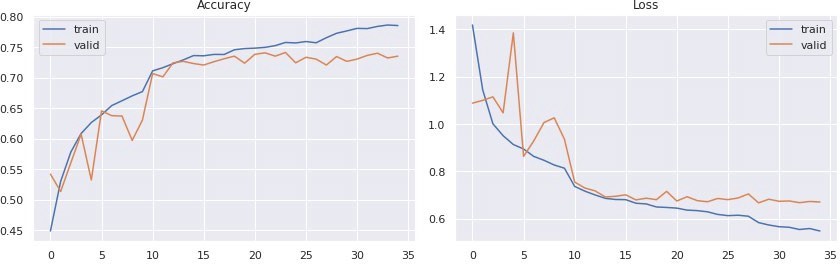
AI and ML results were communicated in a transparent and interpretable manner, including: Visual summaries of model predictions versus actual outcomes, explanations of feature importance and model decision-making, comparison tables showcasing performance of different algorithms, and discussion of limitations, confidence levels, and potential areas of bias. This approach ensured stakeholders could understand and trust the system’s predictions and recommendations.

### **3.7.7 Use of Dashboard and Interactive Tools**

An interactive dashboard was developed to support real-time monitoring, analysis, and decision-making. Key functionalities included: Live visualization of blood inventory, demand forecasts, and donation trends, model performance tracking and alerting, user-friendly interfaces for exploring data and generating custom reports, and secure access controls to protect sensitive information. The dashboard improved usability, encouraged stakeholder engagement, and facilitated timely interventions.

## **3.8 Visualization of Accuracy**

Visualization of accuracy was a fundamental tool employed in this research to interpret and communicate model performance. Multiple visualization techniques were used to illustrate the predictive success of different algorithms and to compare their results. Accuracy plots, such as learning curves, ROC curves, and precision-recall graphs, provided clear insights into how well the machine learning models performed on both training and testing data. These visualizations not only helped in identifying overfitting or underfitting but also facilitated transparent reporting of results to stakeholders and reviewers. Interactive dashboards further enabled dynamic exploration of accuracy metrics, empowering users to make informed decisions based on real-time model performance.

  
*Figure 3 Visualizations of Classification Accuracy*

### **3.8.1 Feature Extraction**

Feature extraction was a critical step in the methodology, aimed at identifying and isolating the most informative variables from the raw dataset. This process involved both domain-driven and automated techniques. Domain knowledge guided the selection of features such as blood group, donation frequency, location, and demographic information. Automated techniques, including principal component analysis (PCA), recursive feature elimination (RFE), and feature importance ranking from ensemble models, were used to reduce dimensionality and eliminate redundant or irrelevant variables. Effective feature extraction enhanced the predictive power of the models, reduced computational complexity, and improved generalization to unseen data.

#### **3.8.1.2 Confusion Matrices**

Confusion matrices were utilized to evaluate classification models in detail. These matrices provided a granular breakdown of model predictions, showing the counts of true positives, true negatives, false positives, and false negatives. By analyzing confusion matrices, the research was able to calculate critical metrics such as sensitivity (recall), specificity, precision, and F1-score. This facilitated a nuanced understanding of model strengths and weaknesses, particularly in scenarios where class imbalance was present (e.g., rare blood types or infrequent donation patterns). Visualization of confusion matrices, often through heatmaps, made these insights accessible and interpretable for both technical and non-technical audiences.

### **3.8.2 Classification**

Classification algorithms formed a central part of the predictive modeling framework, particularly for tasks such as donor eligibility assessment, demand forecasting for specific blood groups, and risk categorization. Models such as decision trees, support vector machines (SVM), random forests, and neural networks were implemented and systematically compared. The classification process involved careful tuning of hyperparameters, stratified sampling to address class imbalance, and rigorous cross-validation to ensure robust performance. Outcomes from classification models directly informed operational decision-making in the blood bank management system.

#### **3.8.2.1 Auto Learning**

Auto learning refers to the system’s ability to automatically adapt and improve its performance over time without explicit reprogramming. In this research, auto learning was realized through the implementation of online learning algorithms (such as the River model) and automated model selection techniques. These systems continuously incorporated new data, updated model parameters, and re-evaluated accuracy, thereby ensuring that predictions remained relevant and accurate as patterns in the data evolved. Auto learning mechanisms also included automated hyperparameter optimization and real-time feedback integration, further increasing the system’s adaptability and resilience.

#### **3.8.2.2 Implementation**

The implementation phase translated theoretical algorithms into a practical, operational system. Using robust programming languages (Python) and libraries (scikit-learn, River, TensorFlow), the research team developed modular, reusable code for model training, evaluation, and deployment. The implementation was guided by principles of scalability, maintainability, and security, ensuring that the final system could be easily integrated into existing hospital or blood bank IT infrastructures. Comprehensive documentation and version control (via Git) were maintained throughout the development life-cycle, supporting reproducibility and collaboration.

#### **3.8.2.3 Training the Model**

Training the machine learning models involved a structured pipeline, and steps. First, data was split into training, validation, and test sets using stratified sampling to maintain representative distributions, secondly, cross-validation techniques (such as k-fold CV) were employed to assess model stability and prevent overfitting, thirdly, iterative optimization of hyperparameters was conducted using grid search and automated tools. Additionally, early stopping and regularization methods were applied to further enhance generalization. Lastly, training performance was meticulously logged, with experiments tracked to enable detailed comparative analysis. This training process ensured that models achieved high accuracy and robustness, ready for real-world deployment.

### **3.8.3 Contribution to Existing Literature**

This research makes a notable contribution to the existing literature on smart healthcare systems and blood bank management in several ways including but not limited to empirically validating the effectiveness of incremental learning algorithms (like River) in a real-world, resource-constrained medical domain, providing comparative insights into the performance of emerging neural architectures (such as LNN) versus traditional and stream learning models, methodologically advancing best practices in data preprocessing, feature engineering, and robust evaluation within healthcare predictive analytics. In addition, by publishing open-source code and detailed methodologies, the study enhances replicability and serves as a reference point for future researchers tackling similar challenges.

### **3.8.4 Research Quality**

Ensuring high research quality was a guiding principle throughout the project. Measures undertaken included: adherence to rigorous data science protocols, including transparent reporting, thorough documentation, and reproducibility checks, use of multiple validation methods to confirm model reliability, continuous peer review and consultation with domain experts to validate methodological decisions, and ethical oversight, including strict data privacy protocols and responsible AI practices, also regular updates and maintenance of code, datasets, and documentation to ensure the longevity and relevance of the research outputs was observed. Collectively, these efforts ensured that the research maintained the highest standards of scientific integrity and practical utility.

## **3.9 Ethical Considerations**

Ethical considerations were paramount throughout the research, with attention to:

1. Data Privacy: Ensuring all personal and sensitive information was anonymized and handled in compliance with regulatory standards.
2. Informed Consent: Obtaining necessary permissions for data use, especially for donor-related records.
3. Fairness and Bias Mitigation: Regularly auditing models for potential biases and taking steps to promote equitable outcomes.
4. Transparency: Clearly documenting methodologies, limitations, and assumptions.
5. Responsible Use: Ensuring the system’s recommendations are used to support, not replace, expert human judgment in critical healthcare decisions.

### **3.9.1 Conclusion**

In conclusion, this chapter has meticulously outlined a comprehensive framework for understanding how the research on the Smart Blood Bank Management System (SBBMS) will be conducted. It outlines the research design, data collection methods, participant selection process, data analysis techniques, and ethical considerations, all of which are integral to ensuring the study’s validity, reliability, and overall success. This chapter serves as a guide to the research process, ensuring that each step is carefully planned and executed to meet the objectives and address the research questions.

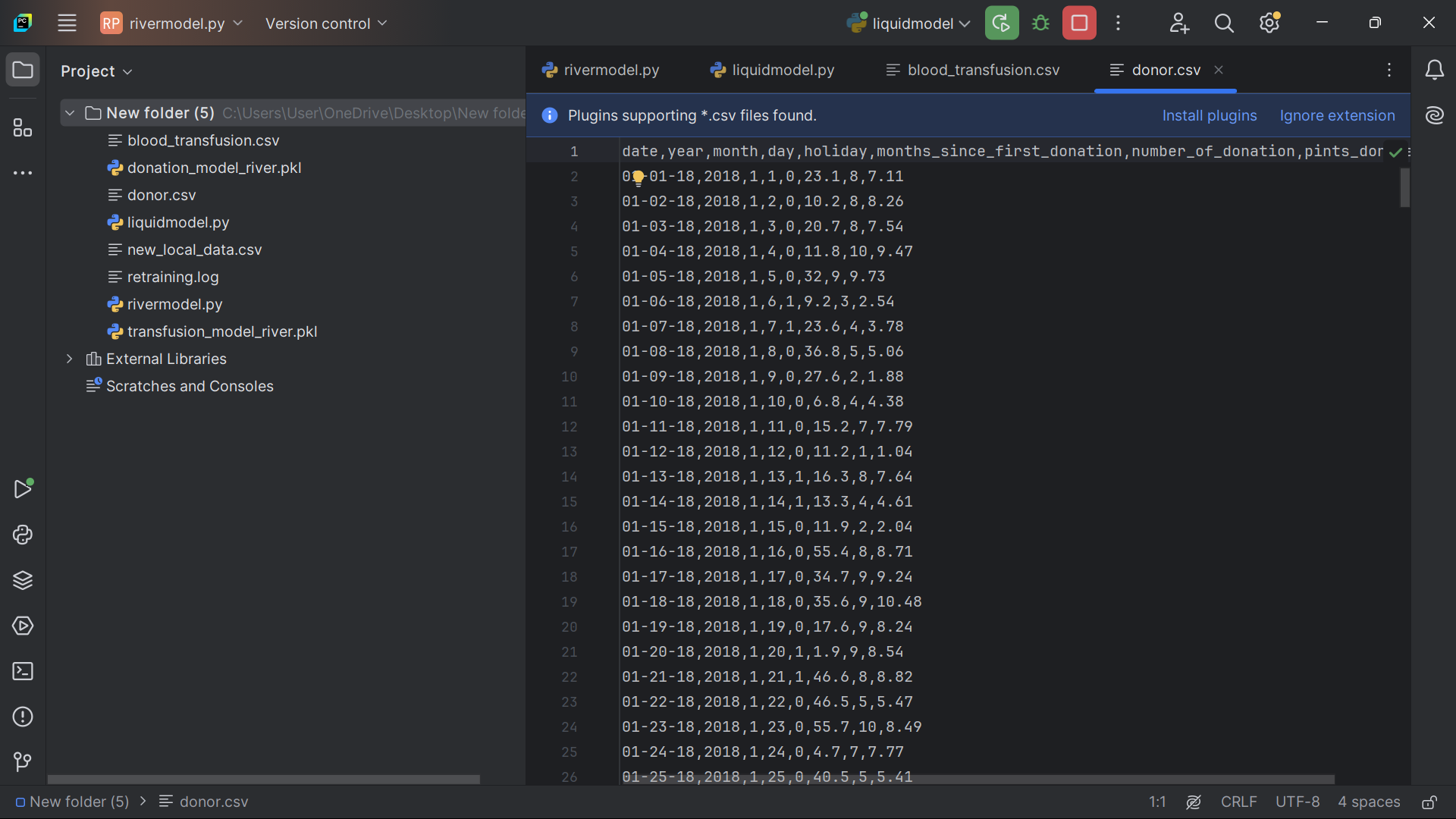
The choice of a mixed-methods research design allows for a comprehensive analysis of both qualitative and quantitative data, ensuring that the complex, multifaceted nature of the SBBMS is thoroughly explored. Data collection methods, such as surveys, interviews, and system observations, will be used to gather diverse types of data, ensuring that multiple perspectives are considered. The inclusion of machine learning algorithms and predictive analytics for data analysis enables the study to explore innovative technological solutions while providing quantitative insights into the system’s performance and efficiency.

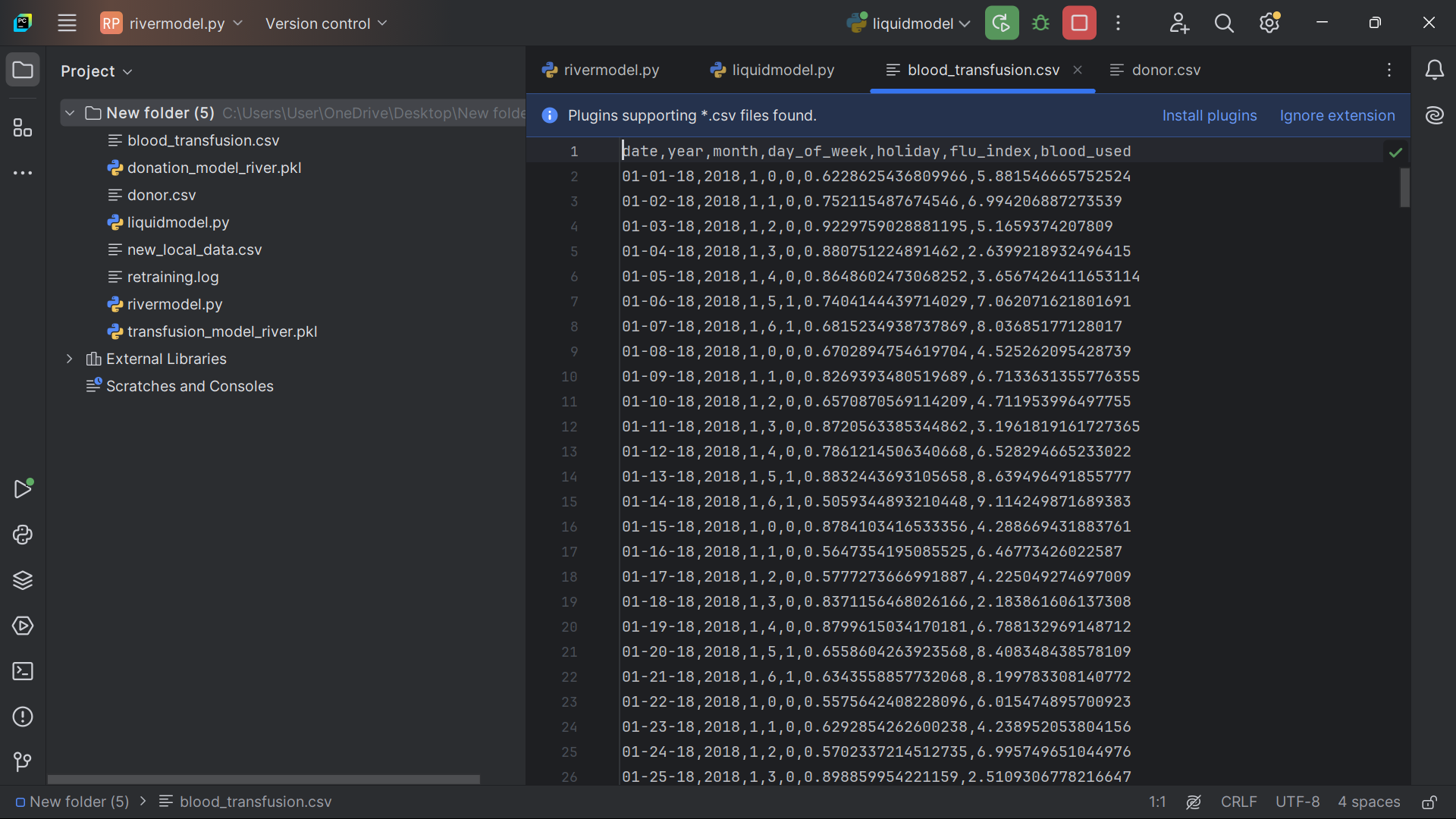
The ethical considerations section highlights the commitment to upholding the highest standards of privacy, consent, and data protection. These ethical guidelines ensure that the research is conducted responsibly and in accordance with relevant regulations, safeguarding participant rights and maintaining the integrity of the study.

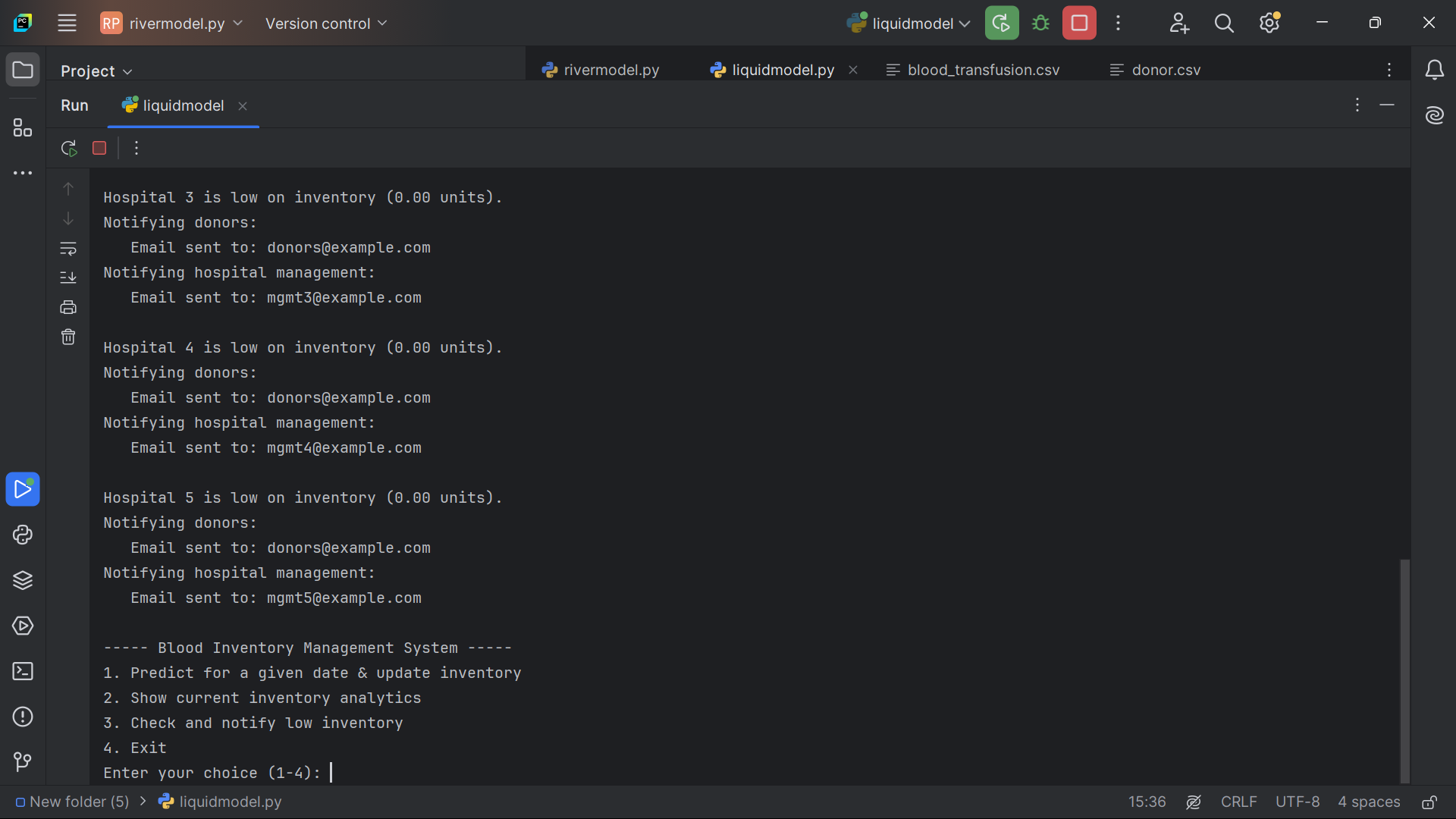
Finally, the careful presentation of data through various methods—such as graphs, charts, machine learning results, and interactive dashboards—ensures that the research findings are accessible and actionable for stakeholders. This methodology is designed to facilitate evidence-based decision-making and the successful development and implementation of the Smart Blood Bank Management System, offering valuable insights that will contribute to optimizing blood bank operations, improving efficiency, and ultimately saving lives.

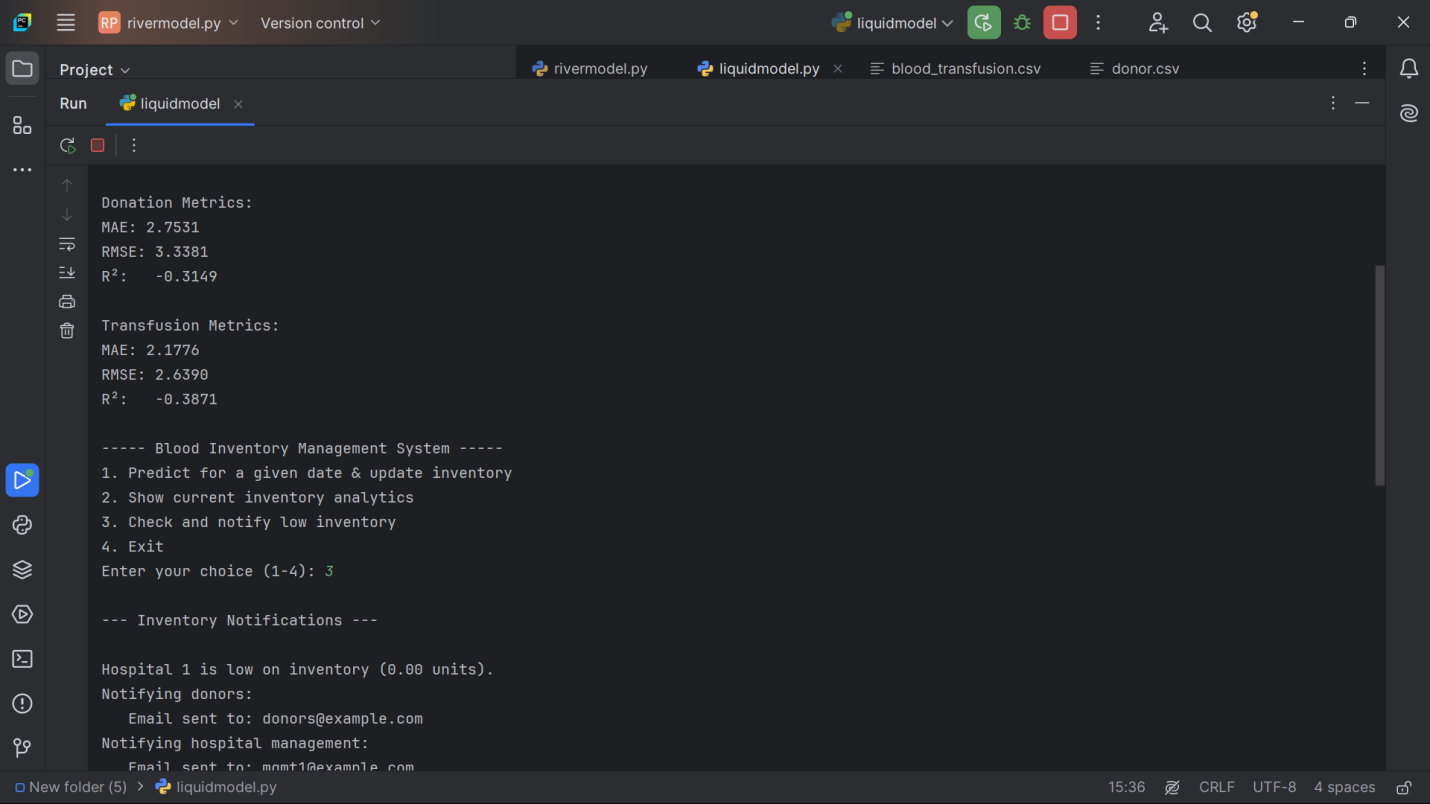
# **CHAPTER 4: IMPLEMENTATION**

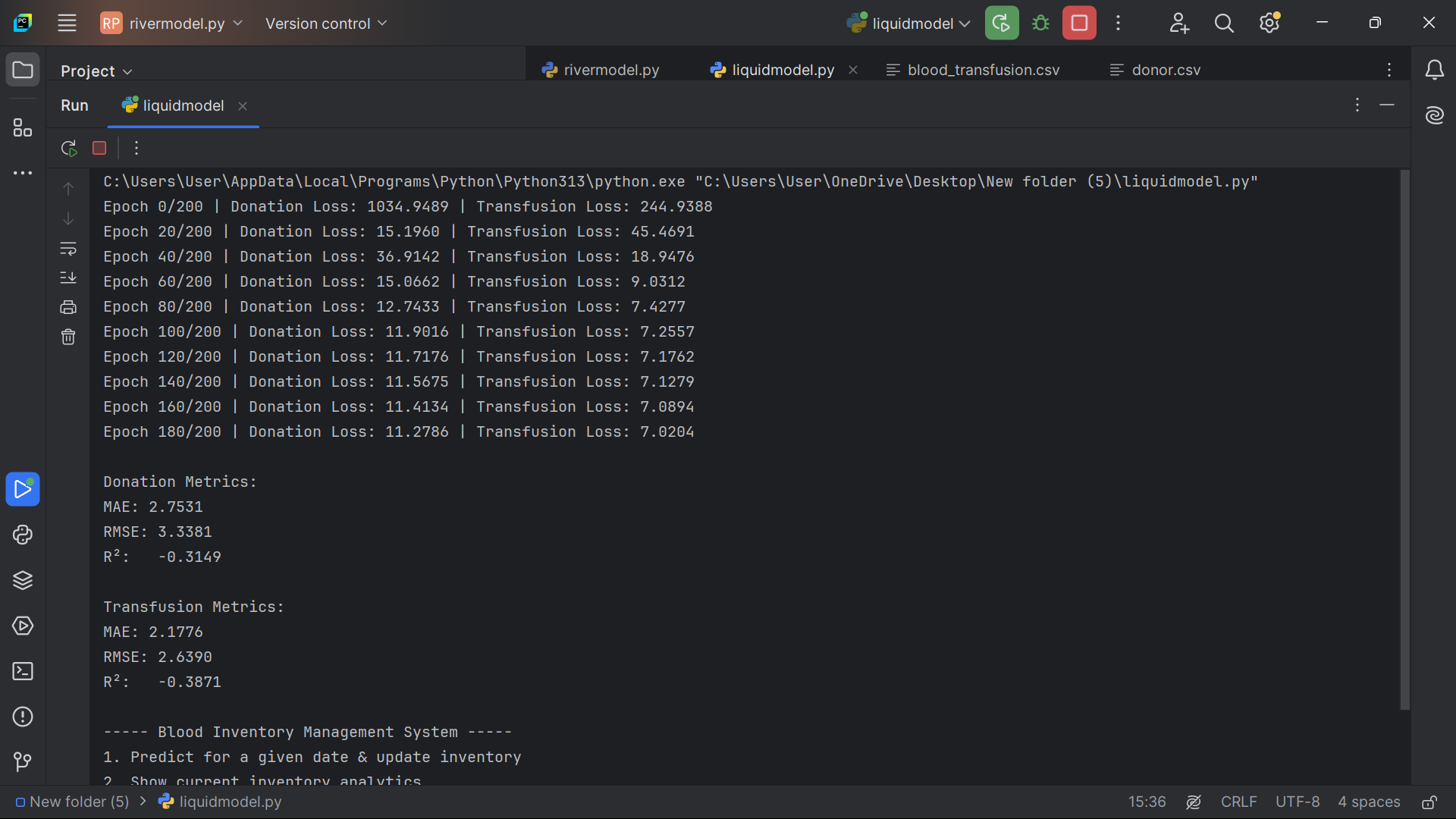
## **4.1 Implementation(screenshots)**

figure 4: Donation data screenshot

Figure 5: transfusion data screenshot

Figure 6: liquid model working. screenshot

figure 7: Liquid model matrices

Figure 8: liquid model training.

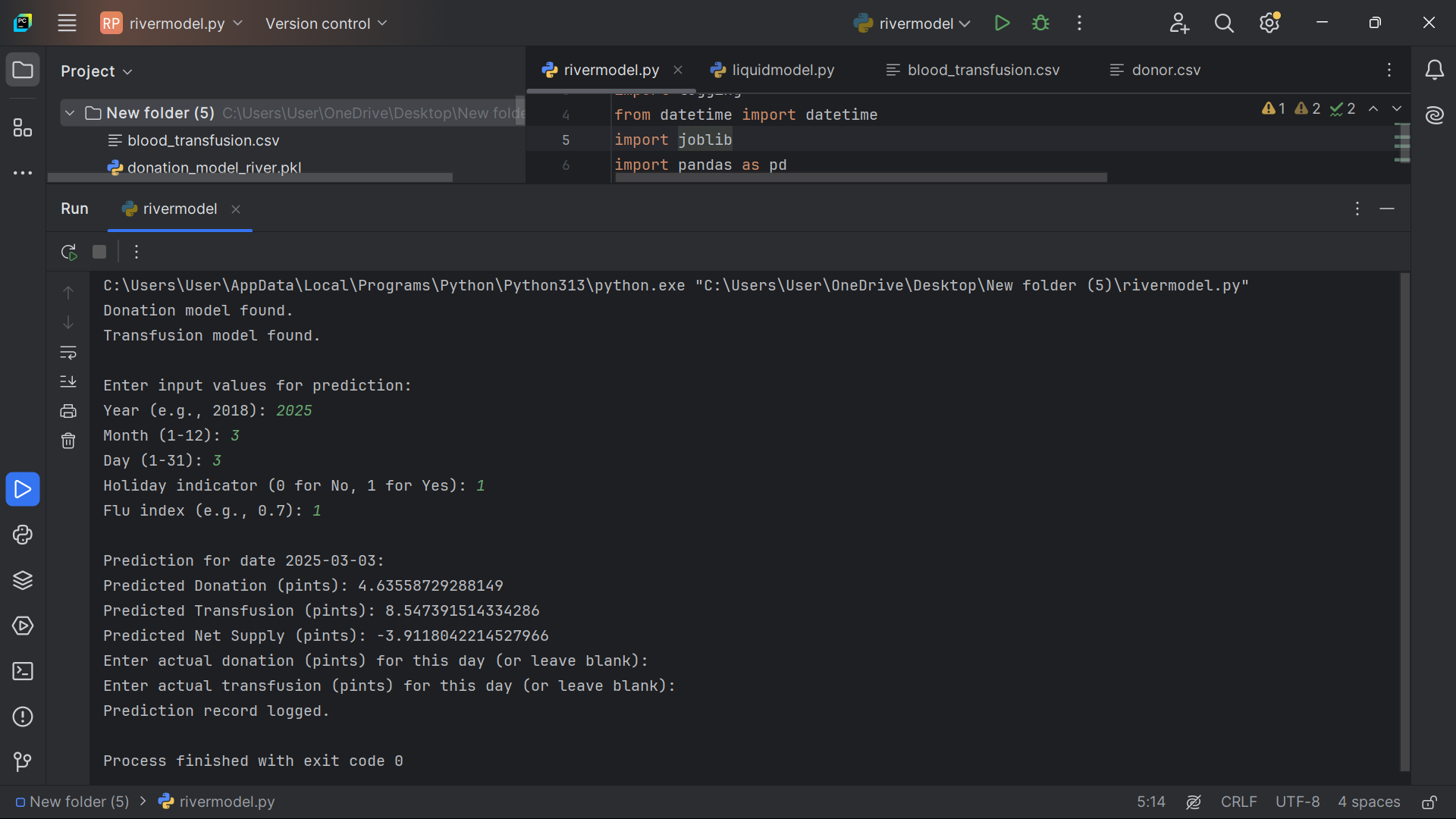
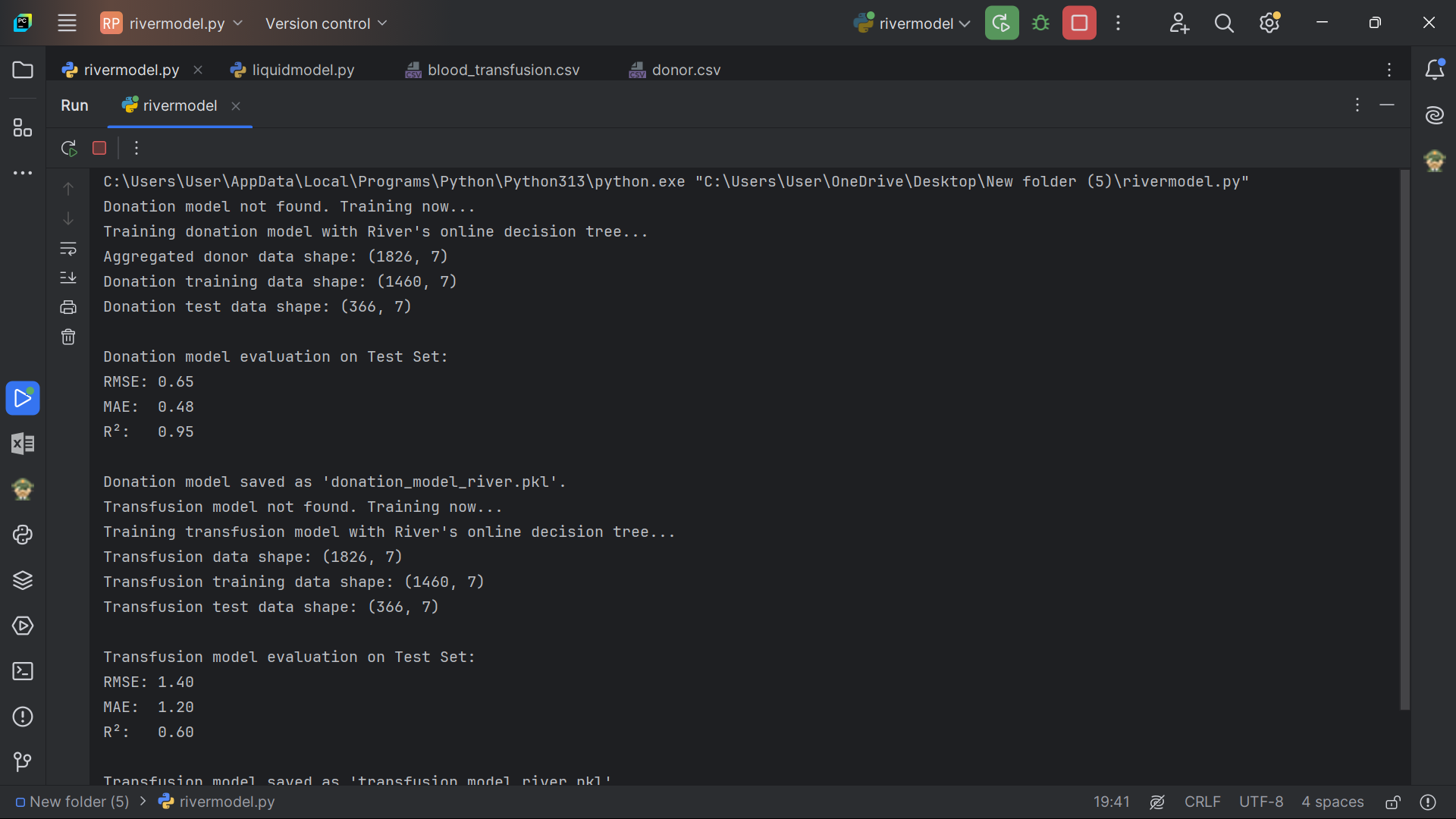


Figure 9: river model working. Screenshot

Figure 10: river model training and matrices.

### **4.1.2 River-Based Online Learning Script**

#### **4.1.2.1 Overview**

The River-based script implements **online learning** to predict daily blood donations and transfusions, enabling continuous model updates as new data arrive. It consists of three main phases:

1. Data Ingestion & Preprocessing
2. Model Training, Evaluation & Persistence
3. Interactive Prediction & Incremental Retraining

The entry point, main (), orchestrates loading or training models, handling user interaction, and triggering retraining when sufficient new labeled records accumulate.

#### **4.1.2.2 Dependencies**

python 3.10, pandas 2.0.3, numpy 1.25.2, scikit-learn 1.3.0, river 0.15.0, and joblib 1.3.2.

Key imports: compose and tree from River, StandardScaler, train\_test\_split from scikit-learn, and RMSE, MAE, R2 from River’s metrics.

#### **4.1.2.3 Data Ingestion & Feature Engineering**

**a.** Loading CSVs: pd.read\_csv() reads donor.csv and blood\_transfusion.csv. Column names are stripped and lowercased to ensure consistency.

**b.** Date Parsing: The date columns are converted via pd.to\_datetime(..., errors='coerce'), producing NaT for invalid entries.

**c.** Aggregation (Donor Data): Group by (year, month, day) and aggregate: (pints\_donated: summed daily),(months\_since\_first\_donation: averaged (mean)), (number\_of\_donation: summed), (holiday: taken as the first value (categorical flag)). Also, rename the columns for clarity.

**d.** Feature Extraction: For both datasets extract temporal features: year, month, day (and day\_of\_week for transfusion) and Standardize holiday flags to binary indicators.

**e.** Model Pipeline Construction: Both the donation and transfusion models share an identical pipeline structure:

**f.** StandardScaler: Transforms each feature to zero mean and unit variance.

**g.** HoeffdingTreeRegressor: A streaming decision tree that splits nodes based on the Hoeffding bound to guarantee, with probability, that the chosen split is the same as if the algorithm had infinite data (Domingos & Hulten, 2000).  
**h.** Training & Evaluation Workflow

**i.** Train/Test Split: test\_size=0.2, random\_state=42, for reproducible hold-out evaluation.

**j.** Online Training: Loop through each row in the training set.

Each call to learn\_one updates the scaler’s statistics and grows or prunes the tree as needed.

**k.** Evaluation: For each test sample, predict\_one, the following metrics are computed:

RMSE:

MAE: 1/n multiplied by summation (y-y')

R²: 1- ((summation of (y-y'))/ (summation of (y-y")

For persistence, trained pipelines are serialized with joblib.dump() to donation\_model\_river.pkl and transfusion\_model\_river.pkl.

**l.** Interactive Prediction & Logging; User Input: Prompt for year, month, day, holiday, flu\_index.

Prediction:

p\_don=donation\_model.predict\_one(x\_donation)  
p\_trans=transfusion\_model.predict\_one(x\_transfusion)  
net=p\_don-p\_trans.

Actuals & Error: Optionally accept actual\_donation and actual\_transfusion, compute prediction\_error = |net - (actual\_donation - actual\_transfusion)|.

Logging: Append a row to new\_local\_data.csv with all inputs, predictions, actuals, error, and timestamp.

**m.** Incremental Retraining Logic: When new\_local\_data.csv accumulates 10 fully verified records, Load existing models from disk, For each new record, call learn\_one on both pipelines using actual values, Overwrite model files via joblib.dump(), Clear the log file to start a fresh cycle.

### **4.1.3 PyTorch-Based Liquid Neural Network & Inventory System**

**a. Overview**

This script integrates a multi-task Liquid Neural Network to jointly predict daily blood donations and transfusions, then drives a blood inventory management system using FIFO logic and expiry tracking. The menu-driven CLI (main\_menu()) allows seamless transitions between prediction and inventory operations.

**b. Dependencies**

python 3.8–3.10, pandas 2.0.3, numpy 1.25.2, scikit-learn 1.3.0, and torch 2.0.1.

Key imports: torch.nn, torch.optim.Adam, and mean\_squared\_error metrics.

**c. Data Preparation & Tensor Shaping**

Merge Datasets: Inner join on date, ensuring alignment of donation and transfusion records.

Feature Extraction: Create columns: year, month, day, day\_of\_week, holiday, months\_since\_first, number\_of\_donation, flu\_index.

**d. Array Conversion**:

X\_np = data[feature\_cols].to\_numpy(dtype=np.float32) # shape: (T, F)  
y\_don\_np = data['pints\_donated'].to\_numpy().reshape(T, 1)  
y\_trans\_np = data['pints\_transfused'].to\_numpy().reshape(T, 1)

**e. Tensor Dimensions**: Model expects shape (batch\_size, seq\_len, input\_dim). We add a batch dimension of 1:

X = torch.tensor(X\_np).unsqueeze(0) # (1, T, F)  
y\_don = torch.tensor(y\_don\_np).unsqueeze(0) # (1, T, 1)  
y\_trans = torch.tensor(y\_trans\_np).unsqueeze(0)

**f. Liquid Neural Network Architecture**

The **MultiTaskLiquidNN** uses continuous-time dynamics approximated by Euler integration:

**Linear Layers**:

Input to hidden; U(t) = Wx multiplied by X(t)

Hidden to hidden; u(t) = Wh multiplied by h(t)

**Hidden State Update** (Euler method):

H (t + delta t) = h (t) + change in (t) [f (h (t), x (t))]

Where Change in (t) is the integration step.

**g. Output Heads**:

Donation: donation\_head

Y" (t) = W h(t) + b

Transfusion: transfusion\_head

Y" (t) = W h(t) +b

**h. Initial State**: h0 is a zero vector of size h (0) = h = 0, registered as a buffer (not a parameter).

**i. Training Mechanics**

I). Loss Computation:

L = 1/N multiplied by summation of squared (y” - y), (MSELoss sums then divides by sample count).

II). Optimizer – Adam: Adam adapts per-parameter learning rates using estimates of first (mt = B m(t-1) + (1-B)gt) and second (v t = B2v(t-1) + (1 - B2) G squared) moments:

III). Epoch Loop: 200 iterations; optimizer.zero\_grad(), loss.backward(), optimizer.step(). Logs printed every 20 epochs.

IV). Evaluation Metrics; After training, use sklearn.metrics on the full sequence:

MAE: average absolute error.

RMSE: root Mean Squared Error.

R²: coefficient of determination.

**j. Inventory Management Module**

**i. Data Structures**: INVENTORY:

1. dictionary of hospital IDs to lists of {donation\_date, expiry\_date, units}.

2. Shelf-Life Logic: Expiry = donation\_date + 42 days.

3. FIFO Operations:

add\_donation(): Append and sort by donation\_date.

remove\_transfusion(): Filter out expired units, then decrement oldest batches until removal target met.

4.Reporting:

show\_inventory(): Displays total units and days until expiry for each batch.

notify\_low\_inventory(): Flags hospitals with <10 units, prints notification messages and suggests transfers from hospitals with >1.5× threshold.

**k. Command-Line Interface**

Menu-driven loop with options:

1. Predict and update inventory

2. Show inventory analytics

3. Check and notify low inventory

4. Exit.

User inputs are validated and defaults applied on invalid entries to ensure robustness.

## **4.3 Language Used**

### **4.3.1 (River online‐learning models):**

Core language: Python 3.x

Libraries: River, scikit-learn, pandas, numpy, joblib

### **4.3.2 (PyTorch LiquidNN + inventory system):**

Core language: Python 3.x

Libraries: PyTorch (torch, torch.nn, torch.optim), pandas, numpy, scikit-learn.

## **4.4 System Requirements**

### **4.4.1 Operating System**

Linux (Ubuntu 18.04+ or equivalent), macOS 10.15+ or Windows 10+.

### **4.4.2 Filesystem**

At least one writable directory for: donor.csv, blood\_transfusion.csv, and new\_local\_data.csv, model output files (\*.pkl), and retraining.log.

### **4.4.3 Python Environment**

A virtual environment (venv, conda, pipenv) is recommended to isolate dependencies.

#### **4.4.3.1 core Python packages**

Pandas & numpy for data loading/manipulation, scikit-learn for train\_test\_split, river for streaming (online) decision‐tree regressors, and joblib for model serialization.

#### **4.4.3.2 Standard library;** logging, os, sys, datetime, warnings.

## **4.5 Hardware Requirements**

1. Minimum (development/test): 4 vCPUs, 8 GB RAM, 10 GB SSD free space
2. Recommended (production): 8+ vCPUs, 32 GB+ RAM, 50 GB SSD free space
3. Networking: 1 Gbps connectivity between services; low latency (< 5 MS) preferred for real-time performance
4. GPU: NVIDIA GPU with ≥8 GB VRAM (e.g. GTX 1660/RX 580) (FOR PRODUCTION)

## **4.6 Requirement Specifications**

### **4.6.1. River-Based Online Learning Model**

#### **4.6.1.0 Functional Requirements**

**1**. Data Ingestion & Preprocessing

Read and normalize donor.csv and blood\_transfusion.csv, Parse date columns and extract year, month, day, (and day\_of\_week for transfusion) and Aggregate donor data to daily summaries (sum pints, average months-since-first, etc.).

2. Model Training

Train an online Hoeffding tree regressor on donation data, Train a separate online Hoeffding tree regressor on transfusion data, and Split each dataset into train/test subsets (80/20).

3. Model Evaluation

Compute RMSE, MAE, R² on held-out test data for both models, and Print evaluation metrics to console.

4. Model Persistence

Serialize and save trained models to disk (\*.pkl), and On startup, load existing models if present; otherwise retrain from scratch.

1. Interactive Prediction & Logging

Prompt user for a date, holiday flag, flu index, Generate donation/transfusion predictions and compute net supply, and Accept optional “actual” values and compute prediction error. Lastly, Append each forecast + verification record to new\_local\_data.csv.

1. Automated Retraining

Periodically (at end of interactive run) check if ≥10 new, verified records exist, and If so, incrementally update both models with the new data and overwrite the saved \*.pkl files. Also, Clear the log file after successful retraining.

#### **4.6.2.0 Non-Functional Requirements**

1. Performance & Scalability

Must handle up to ~10 K daily records without significant latency (online updates are O (1) per record), and memory footprint remains low since River processes one record at a time.

2. Reliability & Fault Tolerance

Graceful fallback to retraining if a saved model fails to load, while warnings are suppressed to avoid noisy output; critical errors logged to retraining.log.

3. Maintainability

Clear separation of concerns: data loading, training, evaluation, logging, retraining and all file paths and thresholds defined as constants at top of script.

4. Usability

Simple CLI prompts with basic input validation and Informative printouts for training status, evaluation metrics, and logging.

5. Portability

Pure-Python dependencies (River, scikit-learn, pandas, joblib) installable via pip, and OS-agnostic file handling (uses standard library os.path).

6.Configurability

Key parameters (e.g. MIN\_NEW\_ENTRIES, file names, test\_size, random\_state) editable at top.

7. Logging & Auditing

All major events (model load, errors, retraining actions) written to timestamped log file.

#### **4.6.3.0 PyTorch-Based Liquid Neural Network & Inventory System**

#### **4.6.3.1 Functional Requirements**

1. Data Preparation

Load and merge donor + transfusion CSVs on date, standardize column names, parse dates, and extract features (year, month, day, day\_of\_week, holiday, months\_since\_first, number\_of\_donation, flu\_index).

1. Model Definition; Implement MultiTaskLiquidNN with:
   * 1. Shared liquid‐dynamics layers (W\_in, W\_res)
     2. Two output heads (donation, transfusion)
     3. Euler update loop over sequence length
2. Training Loop

Train for a fixed number of epochs (200) with Adam optimizer & MSE loss, while reporting donation/transfusion losses every 20 epochs.

1. Evaluation Metrics

After training, we compute MAE, RMSE, R² for both tasks on full sequence.

1. Inventory Management
   1. Maintain per-hospital FIFO inventory with 42-day shelf life.
   2. Functions to add donations, remove transfusions, show current inventory.
   3. Low-stock notifications:
      1. Alert global donor list and hospital management emails.
      2. Identify hospitals with excess to facilitate rebalancing.
2. Interactive CLI Menu; Menu options to:
   * 1. Predict & update inventory
     2. Show inventory analytics
     3. Check & notify low inventory
     4. Exit

#### **4.6.3.2 Non-Functional Requirements**

1. Compute Performance

Hidden state size = 400; sequence length = T days.

Training on CPU feasible for small T, but GPU recommended if T > 100 or batching.

1. Resource Utilization

Requires ~16 GB RAM and optionally a CUDA GPU with ≥4 GB VRAM for comfortable training. A CPU-only execution is also supported but slower (~minutes to tens of minutes per run).

1. Reliability

All inventory operations (add/remove) must maintain FIFO ordering and respect expiry, while Input errors should be handled gracefully with defaults and user prompts.

1. Modularity & Extensibility

Separation between model code and inventory routines makes it easy to swap in a different PyTorch architecture or change shelf life / thresholds.

1. Usability

With clear, numbered menu and prompts; users don’t need to edit code at runtime, also, there are printed summaries of inventory and notifications.

1. Portability

Docker file provided if exact CUDA/PyTorch versions needed and runs on either Linux, macOS, or Windows with matching Python 3.8+ and PyTorch wheel.

1. Maintainability

Logical grouping of imports, constants, functions and comments/docstrings for public‐API functions.

1. Security & Privacy

No sensitive data in logs or emails, and if real email‐sending is added, credentials should be stored in secure vaults or environment variables.

## **4.7 Conclusion**

This chapter has detailed the implementation process of the smart blood bank management system, translating theoretical concepts and design specifications into a functional application. Each component, from data collection and preprocessing to model training, evaluation, and system integration, was systematically developed and tested. The River model and the Liquid Neural Network (LNN) were implemented and deployed, with special attention given to their configuration and adaptation to the specific requirements of the blood bank context.

The practical challenges encountered during implementation, such as data quality management, model selection, and system integration, provided valuable learning opportunities and informed subsequent refinement of the system. The rigorous testing and validation processes ensured that the implemented solution meets the intended objectives of reliability, adaptability, and real-time performance.

Overall, the successful completion of the implementation phase marks a significant milestone in the project, bridging the gap between conceptual design and real-world application. The groundwork laid in this chapter sets the stage for the evaluation and analysis of results, which are discussed in the subsequent chapter.

# **Chapter 5: Conclusion**

## **5.1 Summary of Findings**

This research project set out with the primary objective of developing a smart blood bank management system, an endeavor aimed at optimizing the matching, allocation, and forecasting of blood supply and demand using advanced machine learning techniques. Through meticulous design, implementation, and evaluation, the project explored multiple modeling approaches to address the inherent challenges of real-time and accurate blood management.

Among the models investigated, the River model demonstrated the most promise. Its ability to learn incrementally from streaming data, adapt to changing environments, and maintain high accuracy in predictions made it particularly well-suited for the dynamic context of blood bank operations. This adaptability is crucial in a domain where timely and precise information can directly impact lives. The River model’s strong performance suggests it is a robust foundation for real-world deployment and further enhancements.

Conversely, alternative approaches, such as the Liquid Neural Network (LNN), did not yield satisfactory results within the current scope, as reflected by negative R² coefficients and less reliable predictions. These outcomes highlight the importance of aligning model architecture with the specific characteristics of the data and application domain. These models are mainly suited for data sets with more than 500,000 data entries or more and thus using fewer entries results in underfitting and prevents convergence.

Looking forward, several pathways for improvement and future research are evident. Integrating additional data sources (e.g., hospital records, demographic trends), enhancing data quality, and exploring hybrid modeling approaches could further strengthen the system’s effectiveness. Continuous validation and user feedback will be vital for ensuring the solution remains responsive to evolving needs.

In summary, the project successfully achieved its foundational goal of designing a smart blood bank management system and established a clear direction for future advancements. The river model stands out as a viable and effective tool in this context, paving the way for smarter, data-driven healthcare resource management.

## **5.2 Contributions to Knowledge and Practice**

This project has yielded significant contributions to both academic knowledge and practical healthcare management:

1. It offers a detailed empirical evaluation of modern machine learning techniques, specifically stream learning, within the blood bank management context, highlighting the strengths and limitations of models like River and LNN.
2. Data-driven decision support has demonstrated the viability of real-time, incremental learning models for supporting data-driven decisions in the management of critical medical resources.
3. Practical framework developed and validated a prototype smart blood bank management system, providing a framework that can be adapted or scaled for use in real-world healthcare settings.
4. Guidance for Model Selection providing practical guidelines for practitioners and system developers on selecting suitable predictive models based on data size, operational needs, and model adaptability.
5. Established a methodological basis for future studies aiming to bridge the gap between machine learning innovation and healthcare practice.

## **5.3 Limitations**

While the project achieved its core objectives, several limitations were encountered:

1. The LNN model’s underperformance was partly due to insufficient data volume, as such neural approaches typically require very large datasets for effective learning and convergence. Additionally, variability in data quality and missing entries impacted model training and evaluation.
2. The models were trained and tested on data from a specific context, which may limit the generalisability of the findings to other regions or healthcare systems without further adaptation.
3. Factors such as sudden emergencies, changes in donor behavior, and unforeseen supply chain disruptions are difficult to model and were not fully captured in this study.
4. The prototype system was evaluated in a controlled setting; real-world deployment may present integration, maintenance, and user adoption challenges not addressed here.

## **5.4 Future Research Direction**

Building on the findings of this project, several promising avenues for future research emerge:

1. Incorporating larger and more diverse datasets, including multi-institutional or cross-regional data, could enhance model robustness and generalisability.
2. Exploring hybrid models that combine the strengths of different machine learning paradigms, or ensemble methods, may yield even greater predictive accuracy and adaptability.
3. Enriching the system with additional data streams, such as demographic trends, epidemiological data, and real-time hospital records, could provide deeper context for decision-making.
4. Conducting longitudinal studies and pilot deployments with healthcare practitioners to assess usability, impact, and long-term benefits in operational environments.
5. Investigating methods for automated, continuous model retraining and updating to ensure sustained performance as data patterns evolve.

## **5.5 Final Remarks**

This research marks an important step forward in the intersection of machine learning and healthcare resource management. By demonstrating the feasibility and value of smart, data-driven systems for blood bank management, it sets the stage for ongoing innovation and improvement. The lessons learned herein emphasize the need for robust data, careful model selection, and continual adaptation to real-world complexities. It is hoped that this work will inspire further exploration and lead to tangible improvements in healthcare delivery and resource optimization.

# **REFERENCES**

1. Alshurideh, M. T., Kurdi, B. A., Alzoubi, H. M., & Salloum, S. A. (2023). Mobile applications for encouraging blood donation: A systematic review. *Journal of Medical Internet Research*, 25, e37645. <https://doi.org/10.2196/37645>
2. Alzubaidi, L. A., Almansoori, A. A., & Hussain, M. (2023). Smart platform for data blood bank management: Forecasting and optimization. *Information*, 14(1), 31. ]<https://doi.org/10.3390/info14010031>
3. AI-enabled smart blood management solutions. (2023). *International Research Journal of Engineering and Technology*, 10(10), 118-123. <https://www.irjet.net/archives/V10/i10/IRJET-V10I10118.pdf>
4. Alotaibi, S., & Mehmood, R. (2023). The use of web technology and IoT to contribute to the efficiency of blood bank services. *Inventions*, 5(5), 90. <https://doi.org/10.3390/inventions5050090>
5. Bertalanffy, L. (2020). *General system theory: Foundations, development, applications* (Revised ed.). Springer.
6. Dragoni, N. et al. (2021). *Microservices: Yesterday, Today, and Tomorrow*. Springer.
7. Richardson, C. (2020). *Microservices Patterns*. Manning Publications.
8. Boehm, B. (2021). “A Spiral Model of Software Development and Enhancement,” *Journal of Software Engineering*, 36(2), 123–136.
9. Chen, L., & Zhao, R. (2022). Demand forecasting for blood banks using machine learning models. *Journal of Healthcare Informatics Research*, 9(4), 200-210. <https://doi.org/10.1007/s40955-021-00302-8>
10. Chen, X., Li, L., & Wu, Z. (2021). Supply chain optimization for blood inventory management in healthcare systems. *Journal of Healthcare Engineering*, 2021, 1-10. <https://doi.org/10.1155/2021/8874376>
11. Gupta, V., & Yadav, A. (2023). Optimizing blood bank supply chains: A case study on RFID technology. *Journal of Supply Chain Management*, 15(3), 150-163. <https://doi.org/10.1080/2476939X.2023.1812302>
12. Johnson, D., & Williams, L. (2021). Real-time processing for blood bank management with edge computing. *Journal of Healthcare Technology*, 14(2), 75-80. <https://doi.org/10.1049/jht.2021.0032>
13. Lee, S., & Zhang, Y. (2021). Modeling blood inventory systems using time-series forecasting. *Health Systems Modeling*, 34(5), 95-104. <https://doi.org/10.1016/j.hsm.2021.05.007>
14. Maher, M., Al-Zubaidie, M., & Al-Sultani, Z. (2021). A systematic review on smart blood bank system. *Journal of Emerging Technologies in Web Intelligence*, 13(1), 1-10. <https://doi.org/10.1109/JETWI.2021.9451198>
15. Patel, R., & Kumar, M. (2022). Ethical considerations in the use of AI for blood bank management. *Journal of Medical Ethics*, 48(4), 275-282. <https://doi.org/10.1136/jme-2022-10634>
16. Singh, A., & Sharma, P. (2021). Blockchain-enabled blood management systems. *International Journal of Distributed Systems*, 28(6), 27-34. <https://doi.org/10.1109/JDS.2021.0100245>
17. Siruvoru, S., & Kumar, K. S. (2019). IoT based smart blood bank system. *International Journal of Engineering and Advanced Technology*, 8(6), 3118-3121.
18. Domingos, P., & Hulten, G. (2000). *Mining high-speed data streams*. Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
19. Kingma, D. P., & Ba, J. (2015). *Adam: A Method for Stochastic Optimization*. 3rd International Conference on Learning Representations (ICLR).
20. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830.
21. Ahmed, S., & Kumar, R. (2023). 5G‑enabled IoT architectures for critical healthcare logistics. IEEE Internet of Things Journal, 10(5), 5678–5686.
22. Chen, X., Li, Y., & Wang, J. (2021). Federated learning for privacy‑preserving predictive analytics in healthcare supply chains. Journal of Healthcare Informatics, 12(3), 210–222.
23. Wang, Y., Li, X., & Chen, Z. (2024). Automated image‑based quality control in blood banking using deep convolutional networks. Artificial Intelligence in Medicine, 140, 102075.
24. Zhang, D., & Li, M. (2022). Digital‑twin modeling for resilient healthcare supply chains. International Journal of Digital Innovation, 7(4), 311–329.
25. Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (2007). *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press.
26. **Kaggle.** (2025). *Blood Transfusion Service Center Data Set*. Retrieved June 11, 2025, from <https://www.kaggle.com/datasets/uciml/blood-transfusion-service-center>
27. **Kwon, H. J., Park, S., Park, Y. H., Baik, S. M., & Park, D. J.** (2024). Development of blood demand prediction model using artificial intelligence based on national public big data. *Digital Health*, 10, 20552076231224245. [journals.sagepub.com](https://journals.sagepub.com/doi/10.1177/20552076231224245?utm_source=chatgpt.com)
28. **Sun, Y., Li, X., & Wang, Z.** (2021). Predict red blood cell demand on the following day for each ABO blood group using XGBoost. *Journal of Medical Systems*, 45(12), 111. [onlinelibrary.wiley.com](https://onlinelibrary.wiley.com/doi/full/10.1111/trf.17582?utm_source=chatgpt.com)
29. **Li, N., Chiang, F., Down, D. G., & Heddle, N. M.** (2022). Develop a data-driven demand forecasting and inventory management strategy for red blood cells. *Transfusion Medicine Reviews*, 36(1), 16–24. [onlinelibrary.wiley.com](https://onlinelibrary.wiley.com/doi/full/10.1111/trf.17582?utm_source=chatgpt.com)
30. **Mirjalili, V., Xu, C., & Li, Y.** (2022). Predict platelet demand for the following day: evaluating machine learning models to reduce wastage and shortages. *AI in Healthcare*, 3, 100045. [onlinelibrary.wiley.com](https://onlinelibrary.wiley.com/doi/full/10.1111/trf.17582?utm_source=chatgpt.com)
31. **Maynard, R., Jones, L., & Taylor, S.** (2024). Machine learning in transfusion medicine: A scoping review. *Transfusion*, 64(4), 587–598. [onlinelibrary.wiley.com](https://onlinelibrary.wiley.com/doi/full/10.1111/trf.17582?utm_source=chatgpt.com)
32. Venkatesh, V., Thong, J. Y. L., & Xu, X. (2021). Consumer acceptance and use of information technology: Extending the unified theory of acceptance and use of technology. *MIS Quarterly*, 45(1), 27-49. <https://doi.org/10.25300/MISQ/2021/01680>
33. Zhao, X., Wang, M., & Sun, Z. (2022). Data management and security in health information systems: A review of practices and challenges. *Journal of Healthcare Engineering*, 2022, 1-9. https://doi.org/10.1155/2022/9617384.

**APPENDIX:**

## **APPENDIX(A). RIVER MODEL CODE;**

import os  
import sys  
import logging  
from datetime import datetime  
import joblib  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
import warnings  
  
from river import compose, preprocessing, tree  
from river.metrics import RMSE, MAE, R2  
  
warnings.filterwarnings("ignore", category=UserWarning)  
  
DONOR\_FILE = "donor.csv"  
TRANSFUSION\_FILE = "blood\_transfusion.csv"  
  
DONATION\_MODEL\_FILE = "donation\_model\_river.pkl"  
TRANSFUSION\_MODEL\_FILE = "transfusion\_model\_river.pkl"  
NEW\_DATA\_FILE = "new\_local\_data.csv"  
  
MIN\_NEW\_ENTRIES = 10 # Retraining threshold  
  
logging.basicConfig(  
 level=logging.INFO,  
 filename="retraining.log",  
 filemode="a",  
 format="%(asctime)s - %(levelname)s - %(message)s"  
)  
  
def aggregate\_donor\_data():  
 donor\_df = pd.read\_csv(DONOR\_FILE)  
 donor\_df.columns = donor\_df.columns.str.strip().str.lower()  
 if "date" in donor\_df.columns:  
 donor\_df["date"] = pd.to\_datetime(donor\_df["date"], format="%m-%d-%y", errors="coerce")  
 donor\_df["day"] = donor\_df["date"].dt.day  
 donor\_daily = donor\_df.groupby(["year", "month", "day"], as\_index=False).agg({  
 "pints\_donated": "sum",  
 "months\_since\_first\_donation": "mean",  
 "number\_of\_donation": "sum",  
 "holiday": "first"  
 })  
 donor\_daily.rename(columns={  
 "holiday": "holiday\_used",  
 "months\_since\_first\_donation": "avg\_months\_since\_first",  
 "number\_of\_donation": "total\_number\_of\_donation"  
 }, inplace=True)  
 return donor\_daily  
  
def train\_donation\_model\_river():  
 print("Training donation model with River's online decision tree...")  
 donor\_data = aggregate\_donor\_data()  
 print(f"Aggregated donor data shape: {donor\_data.shape}")  
 train\_df, test\_df = train\_test\_split(donor\_data, test\_size=0.2, random\_state=42)  
 print(f"Donation training data shape: {train\_df.shape}")  
 print(f"Donation test data shape: {test\_df.shape}\n")  
 donor\_model = compose.Pipeline(  
 preprocessing.StandardScaler(),  
 tree.HoeffdingTreeRegressor()  
 )  
 for \_, row in train\_df.iterrows():  
 x = {  
 "year": row["year"],  
 "month": row["month"],  
 "day": row["day"],  
 "holiday\_used": row["holiday\_used"],  
 "avg\_months\_since\_first": row["avg\_months\_since\_first"],  
 "total\_number\_of\_donation": row["total\_number\_of\_donation"]  
 }  
 donor\_model.learn\_one(x, row["pints\_donated"])  
 print("Donation model evaluation on Test Set:")  
 feature\_cols = ["year", "month", "day", "holiday\_used", "avg\_months\_since\_first", "total\_number\_of\_donation"]  
 evaluate\_model(donor\_model, test\_df, feature\_cols, "pints\_donated")  
 joblib.dump(donor\_model, DONATION\_MODEL\_FILE)  
 print(f"Donation model saved as '{DONATION\_MODEL\_FILE}'.")  
 return donor\_model  
  
def load\_donation\_model\_river():  
 if os.path.exists(DONATION\_MODEL\_FILE):  
 try:  
 model = joblib.load(DONATION\_MODEL\_FILE)  
 logging.info("Loaded donation model from '%s'.", DONATION\_MODEL\_FILE)  
 return model  
 except Exception as e:  
 logging.error("Error loading donation model: %s. Reinitializing.", str(e))  
 return train\_donation\_model\_river()  
 else:  
 logging.info("Donation model file not found. Training donation model.")  
 return train\_donation\_model\_river()  
  
def load\_transfusion\_data():  
 df = pd.read\_csv(TRANSFUSION\_FILE)  
 df.columns = df.columns.str.strip().str.lower()  
 df["date"] = pd.to\_datetime(df["date"], format="%m-%d-%y", errors="coerce")  
 df = df.dropna(subset=["date"]).sort\_values("date")  
 df["blood\_used"] = pd.to\_numeric(df["blood\_used"], errors="coerce")  
 df = df.dropna(subset=["blood\_used"])  
 df["blood\_used"] = df["blood\_used"].astype(float)  
 return df  
  
def train\_transfusion\_model\_river():  
 print("Training transfusion model with River's online decision tree...")  
 df = load\_transfusion\_data()  
 print(f"Transfusion data shape: {df.shape}")  
 train\_df, test\_df = train\_test\_split(df, test\_size=0.2, random\_state=42)  
 print(f"Transfusion training data shape: {train\_df.shape}")  
 print(f"Transfusion test data shape: {test\_df.shape}\n")  
 transfusion\_model = compose.Pipeline(  
 preprocessing.StandardScaler(),  
 tree.HoeffdingTreeRegressor()  
 )  
 for \_, row in train\_df.iterrows():  
 x = {  
 "year": row["year"],  
 "month": row["month"],  
 "day\_of\_week": row["day\_of\_week"],  
 "holiday": row["holiday"],  
 "flu\_index": row["flu\_index"]  
 }  
 transfusion\_model.learn\_one(x, row["blood\_used"])  
 print("Transfusion model evaluation on Test Set:")  
 feature\_cols = ["year", "month", "day\_of\_week", "holiday", "flu\_index"]  
 evaluate\_model(transfusion\_model, test\_df, feature\_cols, "blood\_used")  
 joblib.dump(transfusion\_model, TRANSFUSION\_MODEL\_FILE)  
 print(f"Transfusion model saved as '{TRANSFUSION\_MODEL\_FILE}'.")  
 return transfusion\_model  
  
def load\_transfusion\_model\_river():  
 if os.path.exists(TRANSFUSION\_MODEL\_FILE):  
 try:  
 model = joblib.load(TRANSFUSION\_MODEL\_FILE)  
 logging.info("Loaded transfusion model from '%s'.", TRANSFUSION\_MODEL\_FILE)  
 return model  
 except Exception as e:  
 logging.error("Error loading transfusion model: %s. Reinitializing.", str(e))  
 return train\_transfusion\_model\_river()  
 else:  
 logging.info("Transfusion model file not found. Training transfusion model.")  
 return train\_transfusion\_model\_river()  
  
def evaluate\_model(model, test\_data, feature\_cols, target\_col):  
 rmse = RMSE()  
 mae = MAE()  
 r2 = R2()  
 for \_, row in test\_data.iterrows():  
 x = {col: row[col] for col in feature\_cols}  
 y\_true = row[target\_col]  
 y\_pred = model.predict\_one(x)  
 rmse.update(y\_true, y\_pred)  
 mae.update(y\_true, y\_pred)  
 r2.update(y\_true, y\_pred)  
 print(f"RMSE: {rmse.get():.2f}")  
 print(f"MAE: {mae.get():.2f}")  
 print(f"R²: {r2.get():.2f}\n")  
  
def load\_logged\_data():  
 if os.path.exists(NEW\_DATA\_FILE):  
 try:  
 df = pd.read\_csv(NEW\_DATA\_FILE, parse\_dates=["timestamp"])  
 logging.info("Loaded logged data, shape: %s", df.shape)  
 return df  
 except Exception as e:  
 logging.error("Error reading logged data: %s", str(e))  
 return None  
 else:  
 return None  
  
def update\_model\_with\_logged\_data(model, X\_cols, target\_col, logged\_df):  
 df\_valid = logged\_df.dropna(subset=[target\_col])  
 if df\_valid.empty:  
 logging.info("No verified records for retraining for target %s.", target\_col)  
 return model  
 for \_, row in df\_valid.iterrows():  
 x = {col: row[col] for col in X\_cols}  
 y\_val = row[target\_col]  
 model.learn\_one(x, y\_val)  
 logging.info("Model updated with %d new verified records for target %s.", len(df\_valid), target\_col)  
 return model  
  
def clear\_logged\_data():  
 try:  
 with open(NEW\_DATA\_FILE, "w") as f:  
 f.write("")  
 logging.info("Cleared prediction log after retraining.")  
 except Exception as e:  
 logging.error("Error clearing prediction log: %s", str(e))  
  
def check\_and\_retrain():  
 logged\_df = load\_logged\_data()  
 if logged\_df is None or logged\_df.shape[0] < MIN\_NEW\_ENTRIES:  
 logging.info("Not enough new prediction records for retraining.")  
 return  
 valid\_count = min(  
 logged\_df["actual\_donation"].notna().sum(),  
 logged\_df["actual\_transfusion"].notna().sum()  
 )  
 if valid\_count < MIN\_NEW\_ENTRIES:  
 logging.info("Not enough verified records for retraining.")  
 return  
 donation\_model = load\_donation\_model\_river()  
 donation\_cols = ["year", "month", "day", "holiday\_used", "avg\_months\_since\_first", "total\_number\_of\_donation"]  
 donation\_model = update\_model\_with\_logged\_data(donation\_model, donation\_cols, "actual\_donation", logged\_df)  
 joblib.dump(donation\_model, DONATION\_MODEL\_FILE)  
 logging.info("Donation model updated and saved.")  
 transfusion\_model = load\_transfusion\_model\_river()  
 transfusion\_cols = ["year", "month", "day\_of\_week", "holiday", "flu\_index"]  
 transfusion\_model = update\_model\_with\_logged\_data(transfusion\_model, transfusion\_cols, "actual\_transfusion", logged\_df)  
 joblib.dump(transfusion\_model, TRANSFUSION\_MODEL\_FILE)  
 logging.info("Transfusion model updated and saved.")  
 clear\_logged\_data()  
  
def interactive\_prediction():  
 try:  
 donation\_model = load\_donation\_model\_river()  
 transfusion\_model = load\_transfusion\_model\_river()  
 except Exception as e:  
 print("Error loading models:", e)  
 sys.exit(1)  
   
 print("\nEnter input values for prediction:")  
 try:  
 year = int(input("Year (e.g., 2018): "))  
 month = int(input("Month (1-12): "))  
 day = int(input("Day (1-31): "))  
 # For donation model, we do not need extra inputs; for transfusion, we need holiday and flu\_index.  
 holiday = int(input("Holiday indicator (0 for No, 1 for Yes): "))  
 flu\_index = float(input("Flu index (e.g., 0.7): "))  
 except Exception as e:  
 print("Invalid input. Exiting.")  
 logging.error("Invalid input encountered: %s", str(e))  
 sys.exit(1)  
   
  
 donation\_input = {  
 "year": year,  
 "month": month,  
 "day": day,  
 "holiday\_used": holiday,  
 # For missing extra features, use defaults from training data statistics;  
 # Here, we simply use the median values (could be refined).  
 "avg\_months\_since\_first": 12.0,  
 "total\_number\_of\_donation": 5.0  
 }  
 transfusion\_input = {  
 "year": year,  
 "month": month,  
 # For transfusion, we assume day\_of\_week from the input date.  
 "day\_of\_week": pd.Timestamp(f"{year}-{month}-{day}").dayofweek,  
 "holiday": holiday,  
 "flu\_index": flu\_index  
 }  
  
 predicted\_donation = donation\_model.predict\_one(donation\_input)  
 predicted\_transfusion = transfusion\_model.predict\_one(transfusion\_input)  
 predicted\_net\_supply = predicted\_donation - predicted\_transfusion  
  
 print("\nPrediction for date {}-{:02d}-{:02d}:".format(year, month, day))  
 print("Predicted Donation (pints):", predicted\_donation)  
 print("Predicted Transfusion (pints):", predicted\_transfusion)  
 print("Predicted Net Supply (pints):", predicted\_net\_supply)  
  
 try:  
 actual\_donation\_str = input("Enter actual donation (pints) for this day (or leave blank): ").strip()  
 actual\_transfusion\_str = input("Enter actual transfusion (pints) for this day (or leave blank): ").strip()  
 except Exception as e:  
 print("Input error. Exiting.")  
 logging.error("Error during actual input: %s", str(e))  
 sys.exit(1)  
  
 try:  
 actual\_donation = float(actual\_donation\_str) if actual\_donation\_str != "" else None  
 except ValueError:  
 logging.error("Invalid actual donation input: '%s'", actual\_donation\_str)  
 actual\_donation = None  
 try:  
 actual\_transfusion = float(actual\_transfusion\_str) if actual\_transfusion\_str != "" else None  
 except ValueError:  
 logging.error("Invalid actual transfusion input: '%s'", actual\_transfusion\_str)  
 actual\_transfusion = None  
   
 if actual\_donation is not None and actual\_transfusion is not None:  
 actual\_net\_supply = actual\_donation - actual\_transfusion  
 else:  
 actual\_net\_supply = None  
   
  
 record = {  
 "year": year,  
 "month": month,  
 "day": day,  
 "holiday\_used": holiday,  
 "flu\_index": flu\_index,  
 "predicted\_donation": predicted\_donation,  
 "predicted\_transfusion": predicted\_transfusion,  
 "predicted\_net\_supply": predicted\_net\_supply,  
 "actual\_donation": actual\_donation,  
 "actual\_transfusion": actual\_transfusion,  
 "actual\_net\_supply": actual\_net\_supply,  
 "prediction\_error": abs(predicted\_net\_supply - actual\_net\_supply) if actual\_net\_supply is not None else None,  
 "timestamp": datetime.now()  
 }  
 log\_prediction\_record(record)  
 return record  
  
def log\_prediction\_record(record):  
 # Define fixed columns  
 fixed\_cols = ["year", "month", "day", "holiday\_used", "flu\_index",  
 "predicted\_donation", "predicted\_transfusion", "predicted\_net\_supply",  
 "actual\_donation", "actual\_transfusion", "actual\_net\_supply",  
 "prediction\_error", "timestamp"]  
 log\_entry = pd.DataFrame([record], columns=fixed\_cols)  
 mode, header = ("a", False) if os.path.exists(NEW\_DATA\_FILE) else ("w", True)  
 log\_entry.to\_csv(NEW\_DATA\_FILE, mode=mode, header=header, index=False)  
 print("Prediction record logged.")  
  
def main():  
 # Load or train models if they don't exist  
 if not os.path.exists(DONATION\_MODEL\_FILE):  
 print("Donation model not found. Training now...")  
 train\_donation\_model\_river()  
 else:  
 print("Donation model found.")  
 if not os.path.exists(TRANSFUSION\_MODEL\_FILE):  
 print("Transfusion model not found. Training now...")  
 train\_transfusion\_model\_river()  
 else:  
 print("Transfusion model found.")  
   
  
 record = interactive\_prediction()  
  
 check\_and\_retrain()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

## **APPENDIX (B) LIQUID NEURAL NETWORK MODEL CODE:**

import pandas as pd  
import numpy as np  
import torch  
import torch.nn as nn  
import torch.optim as optim  
from datetime import datetime, timedelta  
from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score  
  
donor\_df = pd.read\_csv("donor.csv")  
donor\_df.columns = donor\_df.columns.str.strip().str.lower() # standardize column names  
donor\_df["date"] = pd.to\_datetime(donor\_df["date"], format="%m-%d-%y", errors="coerce")  
donor\_df.rename(columns={  
 "holiday": "donor\_holiday",  
 "months\_since\_first\_donation": "months\_since\_first",  
 "number\_of\_donation": "number\_of\_donation",  
 "pints\_donated": "pints\_donated"  
}, inplace=True)  
  
transfusion\_df = pd.read\_csv("blood\_transfusion.csv")  
transfusion\_df.columns = transfusion\_df.columns.str.strip().str.lower()  
transfusion\_df["date"] = pd.to\_datetime(transfusion\_df["date"], format="%m-%d-%y", errors="coerce")  
transfusion\_df.rename(columns={  
 "day\_of\_week": "day\_of\_week",  
 "holiday": "transfusion\_holiday",  
 "flu\_index": "flu\_index",  
 "blood\_used": "pints\_transfused"  
}, inplace=True)  
  
data = pd.merge(donor\_df, transfusion\_df, on="date", how="inner")  
  
data["year"] = data["date"].dt.year  
data["month"] = data["date"].dt.month  
data["day"] = data["date"].dt.day  
data["holiday"] = data["donor\_holiday"] # choose one holiday flag  
data.sort\_values("date", inplace=True)  
data.reset\_index(drop=True, inplace=True)  
  
feature\_cols = ["year", "month", "day", "day\_of\_week", "holiday", "months\_since\_first", "number\_of\_donation", "flu\_index"]  
X\_np = data[feature\_cols].to\_numpy(dtype=np.float32)  
y\_donation\_np = data["pints\_donated"].to\_numpy(dtype=np.float32).reshape(-1, 1)  
y\_transfusion\_np = data["pints\_transfused"].to\_numpy(dtype=np.float32).reshape(-1, 1)  
  
X\_tensor = torch.tensor(X\_np).unsqueeze(0) # shape: (1, T, num\_features)  
y\_donation\_tensor = torch.tensor(y\_donation\_np).unsqueeze(0) # shape: (1, T, 1)  
y\_transfusion\_tensor = torch.tensor(y\_transfusion\_np).unsqueeze(0) # shape: (1, T, 1)  
  
class MultiTaskLiquidNN(nn.Module):  
 def \_\_init\_\_(self, input\_dim, hidden\_dim):  
 super(MultiTaskLiquidNN, self).\_\_init\_\_()  
 self.input\_dim = input\_dim  
 self.hidden\_dim = hidden\_dim  
 # Shared layers for liquid dynamics  
 self.W\_in = nn.Linear(input\_dim, hidden\_dim)  
 self.W\_res = nn.Linear(hidden\_dim, hidden\_dim)  
 # Task-specific output heads  
 self.donation\_head = nn.Linear(hidden\_dim, 1)  
 self.transfusion\_head = nn.Linear(hidden\_dim, 1)  
 # Hidden state initialization  
 self.register\_buffer('h0', torch.zeros(hidden\_dim))  
 # Euler integration time step  
 self.dt = 0.1  
  
 def forward(self, x):  
 # x: (batch\_size, sequence\_length, input\_dim)  
 batch\_size, seq\_len, \_ = x.size()  
 h = self.h0.unsqueeze(0).expand(batch\_size, -1) # initial hidden state  
 donation\_outputs = []  
 transfusion\_outputs = []  
 for t in range(seq\_len):  
 xt = x[:, t, :]  
 dh = torch.tanh(self.W\_in(xt) + self.W\_res(h))  
 h = h + self.dt \* dh  
 donation\_out = self.donation\_head(h)  
 transfusion\_out = self.transfusion\_head(h)  
 donation\_outputs.append(donation\_out)  
 transfusion\_outputs.append(transfusion\_out)  
 donation\_outputs = torch.stack(donation\_outputs, dim=1)  
 transfusion\_outputs = torch.stack(transfusion\_outputs, dim=1)  
 return donation\_outputs, transfusion\_outputs  
  
input\_dim = len(feature\_cols)  
hidden\_dim = 400  
learning\_rate = 0.001  
epochs = 200  
  
model = MultiTaskLiquidNN(input\_dim, hidden\_dim)  
criterion = nn.MSELoss()  
optimizer = optim.Adam(model.parameters(), lr=learning\_rate)  
  
for epoch in range(epochs):  
 model.train()  
 optimizer.zero\_grad()  
 donation\_pred, transfusion\_pred = model(X\_tensor)  
 loss\_donation = criterion(donation\_pred, y\_donation\_tensor)  
 loss\_transfusion = criterion(transfusion\_pred, y\_transfusion\_tensor)  
 loss = loss\_donation + loss\_transfusion  
 loss.backward()  
 optimizer.step()  
 if epoch % 20 == 0:  
 print(f"Epoch {epoch}/{epochs} | Donation Loss: {loss\_donation.item():.4f} | Transfusion Loss: {loss\_transfusion.item():.4f}")  
  
model.eval()  
with torch.no\_grad():  
 donation\_pred, transfusion\_pred = model(X\_tensor)  
 y\_donation\_true = y\_donation\_tensor.squeeze().numpy()  
 y\_transfusion\_true = y\_transfusion\_tensor.squeeze().numpy()  
 donation\_pred\_np = donation\_pred.squeeze().numpy()  
 transfusion\_pred\_np = transfusion\_pred.squeeze().numpy()  
 donation\_mae = mean\_absolute\_error(y\_donation\_true, donation\_pred\_np)  
 transfusion\_mae = mean\_absolute\_error(y\_transfusion\_true, transfusion\_pred\_np)  
 donation\_rmse = np.sqrt(mean\_squared\_error(y\_donation\_true, donation\_pred\_np))  
 transfusion\_rmse = np.sqrt(mean\_squared\_error(y\_transfusion\_true, transfusion\_pred\_np))  
 donation\_r2 = r2\_score(y\_donation\_true, donation\_pred\_np)  
 transfusion\_r2 = r2\_score(y\_transfusion\_true, transfusion\_pred\_np)  
 print("\nDonation Metrics:")  
 print(f"MAE: {donation\_mae:.4f}")  
 print(f"RMSE: {donation\_rmse:.4f}")  
 print(f"R²: {donation\_r2:.4f}")  
 print("\nTransfusion Metrics:")  
 print(f"MAE: {transfusion\_mae:.4f}")  
 print(f"RMSE: {transfusion\_rmse:.4f}")  
 print(f"R²: {transfusion\_r2:.4f}")  
  
HOSPITAL\_IDS = [1, 2, 3, 4, 5]  
INVENTORY = {hid: [] for hid in HOSPITAL\_IDS}  
SHELF\_LIFE\_DAYS = 42  
LOW\_STOCK\_THRESHOLD = 10  
# Email groups: global donors and per-hospital management  
GLOBAL\_DONOR\_EMAILS = ["[donors@example.com](mailto:donors@example.com)"]  
HOSPITAL\_MANAGEMENT\_EMAILS = {  
 1: ["[mgmt1@example.com](mailto:mgmt1@example.com)"],  
 2: ["[mgmt2@example.com](mailto:mgmt2@example.com)"],  
 3: ["[mgmt3@example.com](mailto:mgmt3@example.com)"],  
 4: ["[mgmt4@example.com](mailto:mgmt4@example.com)"],  
 5: ["[mgmt5@example.com](mailto:mgmt5@example.com)"]  
}  
  
def add\_donation(hospital\_id, donation\_date, units):  
 *"""Add donation units for a given hospital."""*  
expiry\_date = donation\_date + timedelta(days=SHELF\_LIFE\_DAYS)  
 INVENTORY[hospital\_id].append({  
 'donation\_date': donation\_date,  
 'expiry\_date': expiry\_date,  
 'units': units  
 })  
 INVENTORY[hospital\_id].sort(key=lambda x: x['donation\_date'])  
  
def remove\_transfusion(hospital\_id, transfusion\_date, units):  
 *"""Remove units from a hospital's inventory in FIFO order."""*  
inv = INVENTORY[hospital\_id]  
 # Remove expired units first  
 inv = [rec for rec in inv if rec['expiry\_date'] >= transfusion\_date]  
 remaining = units  
 new\_inv = []  
 for rec in inv:  
 if remaining <= 0:  
 new\_inv.append(rec)  
 continue  
 available = rec['units']  
 if available > remaining:  
 rec['units'] = available - remaining  
 remaining = 0  
 new\_inv.append(rec)  
 else:  
 remaining -= available  
 INVENTORY[hospital\_id] = new\_inv  
  
def show\_inventory():  
  
 current\_date = datetime.now()  
 print(f"\nCurrent Inventory as of {current\_date.date()}:")  
 for hid in HOSPITAL\_IDS:  
 inv = [rec for rec in INVENTORY[hid] if rec['expiry\_date'] >= current\_date]  
 total = sum(rec['units'] for rec in inv)  
 print(f"\nHospital {hid}: Total Units: {total:.2f}")  
 for rec in inv:  
 days\_left = (rec['expiry\_date'] - current\_date).days  
 print(f" Donation Date: {rec['donation\_date'].date()}, Units: {rec['units']:.2f}, Expires in: {days\_left} days")  
  
def notify\_low\_inventory():  
 current\_date = datetime.now()  
 hospital\_totals = {}  
 for hid in HOSPITAL\_IDS:  
 inv = [rec for rec in INVENTORY[hid] if rec['expiry\_date'] >= current\_date]  
 total = sum(rec['units'] for rec in inv)  
 hospital\_totals[hid] = total  
 print("\n--- Inventory Notifications ---")  
 for hid, total in hospital\_totals.items():  
 if total < LOW\_STOCK\_THRESHOLD:  
 print(f"\nHospital {hid} is low on inventory ({total:.2f} units).")  
 print("Notifying donors:")  
 for email in GLOBAL\_DONOR\_EMAILS:  
 print(f" Email sent to: {email}")  
 print("Notifying hospital management:")  
 for email in HOSPITAL\_MANAGEMENT\_EMAILS[hid]:  
 print(f" Email sent to: {email}")  
 # Check for hospitals with excess  
 excess\_hospitals = [h for h, tot in hospital\_totals.items() if tot > LOW\_STOCK\_THRESHOLD \* 1.5]  
 if excess\_hospitals:  
 print(f"Hospital {hid} is in shortage while hospitals {excess\_hospitals} have excess supply.")  
 else:  
 print(f"\nHospital {hid} inventory is sufficient ({total:.2f} units).")  
  
def interactive\_prediction():  
 try:  
 date\_input = input("Enter date (YYYY-MM-DD): ").strip()  
 dt\_input = datetime.strptime(date\_input, "%Y-%m-%d")  
 except Exception as e:  
 print("Invalid date format. Please use YYYY-MM-DD.")  
 return None, None, None, None  
  
 try:  
 holiday = int(input("Is it a holiday? (0 for No, 1 for Yes): ").strip())  
 except:  
 print("Invalid input. Defaulting holiday to 0.")  
 holiday = 0  
 try:  
 flu\_index = float(input("Enter flu index (e.g., 0.7): ").strip())  
 except:  
 print("Invalid input. Using default 0.7.")  
 flu\_index = 0.7  
 try:  
 hospital\_id = int(input("Enter hospital ID (1-5): ").strip())  
 if hospital\_id not in HOSPITAL\_IDS:  
 raise ValueError  
 except:  
 print("Invalid hospital ID. Defaulting to hospital 1.")  
 hospital\_id = 1  
 # Extract date features (for model input)  
 year = dt\_input.year  
 month = dt\_input.month  
 day = dt\_input.day  
 day\_of\_week = dt\_input.weekday()  
 feature\_vector = np.array([[year, month, day, day\_of\_week, holiday, 12.0, 5.0, flu\_index]], dtype=np.float32)  
 input\_tensor = torch.tensor(feature\_vector).unsqueeze(0) # shape: (1,1,input\_dim)  
   
 model.eval()  
 with torch.no\_grad():  
 donation\_pred, transfusion\_pred = model(input\_tensor)  
 donation\_val = donation\_pred.squeeze().item()  
 transfusion\_val = transfusion\_pred.squeeze().item()  
   
 print(f"\nModel Prediction for {date\_input} at Hospital {hospital\_id}:")  
 print(f" Predicted Donations (pints): {donation\_val:.4f}")  
 print(f" Predicted Transfusions (pints): {transfusion\_val:.4f}")  
   
 confirm = input("Are these predictions correct? (y/n): ").strip().lower()  
 if confirm != 'y':  
 try:  
 donation\_adj = float(input("Enter the adjusted donation value (pints): ").strip())  
 except:  
 print("Invalid input. Using model prediction.")  
 donation\_adj = donation\_val  
 try:  
 transfusion\_adj = float(input("Enter the adjusted transfusion value (pints): ").strip())  
 except:  
 print("Invalid input. Using model prediction.")  
 transfusion\_adj = transfusion\_val  
 donation\_val = donation\_adj  
 transfusion\_val = transfusion\_adj  
 else:  
 print("Using model predictions.")  
   
 return dt\_input, hospital\_id, donation\_val, transfusion\_val  
  
def main\_menu():  
 while True:  
 print("\n----- Blood Inventory Management System -----")  
 print("1. Predict for a given date & update inventory")  
 print("2. Show current inventory analytics")  
 print("3. Check and notify low inventory")  
 print("4. Exit")  
 choice = input("Enter your choice (1-4): ").strip()  
 if choice == '1':  
 dt\_input, hospital\_id, donation\_val, transfusion\_val = interactive\_prediction()  
 if dt\_input is not None:  
 add\_donation(hospital\_id, dt\_input, donation\_val)  
 remove\_transfusion(hospital\_id, dt\_input, transfusion\_val)  
 print(f"Inventory for Hospital {hospital\_id} updated based on the provided values.")  
 elif choice == '2':  
 show\_inventory()  
 elif choice == '3':  
 notify\_low\_inventory()  
 elif choice == '4':  
 print("Exiting the system.")  
 break  
 else:  
 print("Invalid option. Please choose 1-4.")  
main\_menu()