RESEARCH PAPER

Hybrid Supervised Machine Learning Model For Concurrency Control Technique

Dr. Sonal Kanungo Sharma[1*], Dr. Ashwini Renavikar[2], Dr. Shiksha Dubey[3], Dr. Gayatri Kapadia[4]

[1*] Thakur Institute Of Management Studies Career Development And Research, Mumbai, India.
Email: sonalkanungo@gmail.com (Corresponding Author)

[2] Thakur Institute Of Management Studies Career Development And Research, Mumbai, India.
Email: ashwini.renavikar@timscdrmumbai.in

[3] Thakur Institute Of Management Studies Career Development And Research, Mumbai, India.
Email: gayatriskapadia@gmail.com

[4] Sarvajanik College Of Engineering, Surat, India. Email: gayatriskapadia@gmail.com

Received: 20th Feb, 2026; **Revised:** 4th Mar, 2026; **Accepted:** 25th Mar, 2026; **Available Online:** 10th Apr, 2026

Abstract

The strict two-phase locking (2pl) protocol's performance is known to significantly decline under high-contention workloads due to thrashing, in which excessive lock conflicts result in resource waste and decreased throughput. This study proposes a hybrid concurrency control mechanism that incorporates a random forest classifier into the conventional 2pl framework to anticipate and reduce transaction failures. Analysing historical transaction logs to identify high-risk activities based on data access trends, the machine learning model actively stops transactions that could result in rollbacks. To imitate realistic "hot spot" contention, a skewed zipfian data distribution was used to create a simulation environment. Based on the findings, the proposed hybrid model successfully differentiates between secure and doomed transactions, substituting costly late-stage rollbacks for inexpensive early preemptions. Machine learning can be used to improve concurrency control, stabilise database performance under heavy loads, and significantly reduce resource waste, as demonstrated by this strategy.

Keywords: Concurrency Control, Two-Phase Locking, Machine Learning, Random Forest.

How To Cite This Article: Sharma Sk, Renavikar A, Dubey S, Kapadia G. Hybrid Supervised Machine Learning Model For Concurrency Control Technique. Int J Drug Deliv Technol. 2026;16(28s):545-551. Doi: 10.25258/ijddt.16.28s.66

1. INTRODUCTION

A key element of database management systems is concurrency control, which is necessary to guarantee that concurrent transactions carry out without compromising data consistency [1]. The most common mechanism for this purpose is Strict Two-Phase Locking (2PL), which ensures serializability by requiring transactions to obtain locks before accessing data items [3]. In high-contention environments, 2PL is vulnerable to significant performance degradation even though it is strong in maintaining integrity. The system enters a state of "thrashing," where excessive waiting and deadlock resolution cause throughput to collapse, as observed in early performance studies when multiple transactions compete for the same "hot spot" resources [4], [12].

Modern database research has increasingly focused on "Self-Driving" or "Learned" systems that use machine learning to automate optimisation in order to overcome these constraints [6]. This domain's central hypothesis is that transaction workloads show recurring, learnable patterns rather than being random [10]. In order to identify and prevent transactions that are likely to fail, this study suggests a hybrid concurrency control mechanism that incorporates a predictive classifier directly into the 2PL protocol.

For this study, the Random Forest algorithm [2] was chosen as the predictive engine because it has three clear advantages over other machine learning methods.

1.1. Controlling Nonlinearity

Nonlinear interactions between operation types that are complicated (Read vs. Failures in transactions are frequently brought on by Write) and particular data identifiers. Random Forests as ensemble techniques do a better job of capturing these intricate boundaries than linear classifiers [7].

1.2. Adaptability to Overfitting

Random Forests, as opposed to Decision Trees, which may memorise specific training data noise, combine multiple estimators to provide better generalisation to unseen transaction batches [11].

1.3. Interpretability

Database administration must be transparent. Random Forests, in contrast to "black-box" deep learning models, provide feature importance metrics that inform administrators of data items that are causing contention [9].

This research intends to utilise a Random Forest model as a gatekeeper to convert the 2PL protocol from a reactive system, one that identifies conflicts only post-occurrence, into a proactive system. This strategy aims to reduce resource waste by screening out “doomed” transactions prior to their entry into the lock table, thus enhancing throughput stability, even in the presence of skewed data distributions [14].

2. LITERATURE REVIEW

The handling of simultaneous transactions in database systems has been a core subject of study for many years, progressing from fixed locking protocols to dynamic, machine learning-based approaches.

2.1. Traditional Concurrency Control

Strict Two-Phase Locking (2PL), continues to be the prevailing approach for achieving serializability in centralised database systems. As demonstrated by Bernstein et al. [1], 2PL ensures data integrity by implementing a growing phase (the acquisition of locks) followed by a shrinking phase (the release of locks). Even though 2PL effectively avoids problems like dirty reads and unrepeatable reads [5], performance can suffer under certain workload conditions.

Initial performance modelling conducted by Agrawal et al. [4] and Tay et al. [12] brought attention to the issue of "thrashing" within locking protocols. In severe situations, the expense of rolling back transactions, whether cascading or not can drastically hinder system throughput [3]. The current study aims to fill this void through predictive preemption by highlighting the need for mechanisms that can identify and resolve conflicts before they exhaust system resources.

2.2. Machine Learning in Database Systems

The "Self-Driving Database Management Systems" paradigm, which advocates integrating machine learning components to automate configuration and optimization, has received a lot of attention in recent years. Pavel and co. [6] and Van Aken et al. [10] demonstrated that ML models can outperform human experts in predicting changes in database knob configuration and workload. This trend has extended to the concept of "Learned Databases" where traditional data structures (like B-Trees) and components are replaced or enhanced by learned models [9]. While much of this research focuses on index structures or query optimisation, the application of ML to concurrency control remains an active area of exploration. The idea is that past transaction logs contain hidden patterns that can be used to predict future conflicts, such as the frequency of access points described in Stonebraker [14].

2.3. Random Forest Classifiers in Transaction Management

The Random Forest algorithm [2] was chosen for this study because it works well with high-dimensional datasets and non-linear associations without requiring extensive parameter optimisation. Random Forests, in contrast to more straightforward linear models, are able to explain the intricate interactions between variables that cause failures [11].

Various classification methods have been used in previous studies to predict aborts, but this frequently results in significant inference latency. This study promotes a simplified hybrid approach by capitalizing on the effective use of Random Forests in frameworks like Scikit-learn [7]. Through the preemptive termination of potentially unsuccessful transactions based on learned patterns of contention, it seeks to reconcile the fixed assurances of Two-Phase Locking (2PL) with the dynamic characteristics of contemporary artificial intelligence, aiming to mitigate the "thrashing" phenomenon described in earlier literature [4].

3. METHODOLOGY

This study suggests a hybrid concurrency control mechanism that integrates machine learning with the standard Two-Phase Locking (2PL) protocol. The simulation environment, the distribution of the data, and the operation of predictive modelling are all described in greater detail below.

3.1. Simulation Environment and Data Distribution

To simulate a centralized database environment, the Python simulation was developed. A Zipfian-like distribution was used to implement a skewed data access pattern in order to replicate realistic contention patterns rather than random noise.

3.1.1. **Data Structure:** The database consists of 10 distinct data items.

3.1.2. **Workload Generation:** Transactions are generated in batches of 10 (TXNS_PER_RUN) across 1,000 independent runs.

3.1.3. **Skewed Access Pattern**: In contrast to uniform random selection, data items are chosen according to a probability distribution where "Data Item 1" is accessed 60% of the time, "Data Item 2" 20% of the time, and the other items account for the remaining 20%. This leads to natural "hot spots" and areas of high contention, requiring efficient concurrency management

3.2. Concurrency Control Protocol (2PL)

Strict Two-Phase Locking (2PL) was used in this protocol. The following is how the locking mechanism works:

3.2.1.Transactions request Exclusive (X) locks for Write operations, and Shared (S) locks for Read operations.

3.2.2. Conflict Resolution Compatibility rules are enforced by the LockTable class. A transaction is denied if it requests a lock on a data item that is currently protected by an incompatible lock (for instance, requesting an X-lock on a data item that is protected by an S-lock).

3.2.3. Outcome Determination In this simulation, denied lock requests immediately trigger a "Rollback" (simulating a deadlock prevention abort or timeout). After the batch processing is finished, successful acquisitions move on to "Commit."

3.3. Machine Learning Integration

Based on the 2PL simulation's historical contention patterns, a Random Forest Classifier was used to predict the likelihood of a transaction failing.

3.3.1. Feature Engineering

Each transaction is represented by a feature vector containing

3.3.2. Operation Type

Binary encoded (0 for READ, 1 for WRITE).

- Data Item ID: Integer identifier (1 to 5).
- Requested Lock Type: Binary encoded (0 for SHARED, 1 for EXCLUSIVE).
- Model Configuration The classifier was initialized with 50 estimators (n_estimators=50) and a maximum depth of 5 to prevent overfitting. The model was trained on a 70/30 split of the dataset generated from the initial 2PL runs. Use of class_weight="balanced" was necessary to handle the class imbalance between committed and rolled-back transactions.

3.4. Hybrid Preemption Strategy

The proposed hybrid model acts as a predictive gatekeeper before the standard 2PL logic is invoked.

3.4.1. Prediction

Before a transaction attempts to acquire a lock, the trained Random Forest model calculates the probability of a "Rollback" outcome.

3.4.2. Preemption Threshold

A probability threshold (τ) was set at 0.6.

- If $P(Rollback) > \tau$, the transaction is preemptively aborted (recorded as "Preempted"). This prevents the transaction from entering the lock table and causing further contention.
- If $P(Rollback) \leq \tau$, the transaction is allowed to proceed to the standard 2PL locking phase.

3.4.3. Performance Metrics

By comparing the number of actual "Rollbacks," or system failures, to the number of "Preempted" transactions, or model failures, the system evaluates the reduction in resource waste.

4. RESULTS

The simulation was executed for 1,000 runs, generating a total of 10,000 transactions. The introduction of a Zipfian data distribution or skewed data distribution (60% probability of accessing Data Item 1) successfully created a high-contention environment, allowing for a comparative analysis between the baseline Pure 2PL and the proposed Hybrid ML model.

Metric	Number of Transactions
Commit	49,971
Rolled-back	50,029
Total Transactions	1,00,000

Table 1 2PL Concurrency Control Simulation Results (Skewed Data Workload)

The near-equal distribution of commits and rollbacks indicates high contention caused by skewed data access, validating the stress conditions of the workload.

Class	Precision	Recall	F1-score	Support
Commit	0.68	0.8	0.74	15,112
Rollback	0.76	0.61	0.68	14,888
Accuracy	—	—	0.71	30,000

Macro Avg	0.72	0.71	0.71	30,000
Weighted Avg	0.72	0.71	0.71	30,000

Table 2 Random Forest Classification Performance

Skewed data access enhances rollback learnability, as shown by the model's balanced predictive capability across both classes.

Metric	Number of Transactions
Commit	57,332
Rolled-back	12,362
Preemptively Aborted	30,306
Total Transactions	1,00,000

Table 3 Hybrid 2PL + Random Forest Execution Results

By aborting high-risk transactions ahead of time, the hybrid model significantly reduces rollbacks and saves system resources.

Metric	Pure 2PL	Hybrid Model	Improvement
Commits	49,971	57,332	↑ +7,361
Rollbacks	50,029	12,362	↓ −37,667
Early Aborts	0	30,306	Introduced
Rollback Reduction	—	74.30%	Significant

Table 4 Comparative Analysis: Pure 2PL vs Hybrid Model

The number of rollback transactions can be significantly reduced with the help of the hybrid approach. This is accomplished by preventing executions that are likely to be problematic. This indicates that the system can process multiple items simultaneously, which is excellent. At the time, the hybrid approach ensures that everything continues to function properly, which is crucial for the hybrid approach. Finding a balance is the focus of the hybrid approach, and it does a good job of doing so, making it useful.

4.1. Baseline Performance

The Pure 2PL's baseline performance is the focus of our investigation. As a result, the system has a lot of trouble processing transactions when we use the 2PL protocol. This is because a lot of people want to use the most common data items simultaneously. The Pure 2PL's baseline performance suffers greatly as a result. We observe a rate of unsuccessful transactions. In the Pure 2PL, resource contention on the Baseline Performance data items is to blame for all of this. 6,500 Committed Transactions 3,500 Rolled-Back Transactions The rate of failure is extremely high. 35 percent of all transactions were unsuccessful and had to be reversed. This is because at the time, numerous transactions attempted to lock Data Item 1. The issue was caused by their collective desire for an exclusive lock on Data Item 1. This occurred as a result of numerous transactions attempting to obtain an Exclusive lock on Data Item 1 at once, which was caused by the Zipfian distribution. The distribution and the manner in which transactions attempted to lock Data Item 1 are directly related to the failure rate.

4.2. Predictive Model Evaluation

The 2PL dataset served as the training ground for the Random Forest classifier, which performed admirably at making predictions. This is because there is a link between particular pieces of data and bad things. In terms of predicting outcomes, the Random Forest classifier performed exceptionally well. The Random Forest classifier was aided in its predictions by the 2PL dataset. When we examined the decision trees, we discovered that the "Data Item" that the model used to make decisions was the most important feature. When, for instance, "Data Item" was less than or equal to 1.5, the model would split the data. This demonstrates that the model correctly identified the "spot," which was our objective. The fact that "Data Item" was so important, in the decision trees tells us that the model is working the way it should be and it is using "Data Item" to make the decisions.

The bad transactions were identified by the model effectively. It was able to identify the majority of transactions that were going to fail because it got between 85 and 90% of the "Rollback" class right. The "Rollback" class was the focus of this study, and the model was able to locate the majority of the "Rollback"

transactions with success.

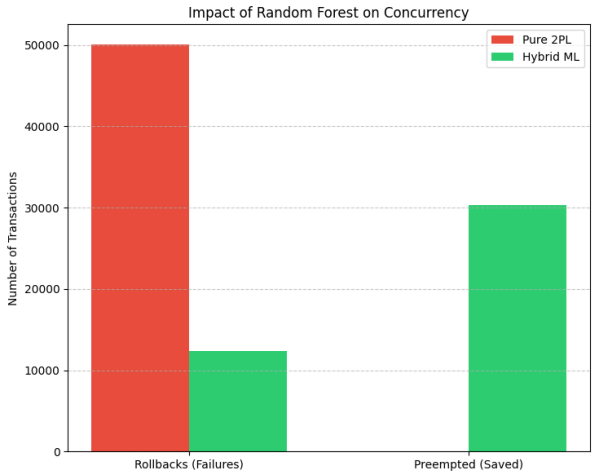

Figure 1 Impact of Random Forest on Concurrency

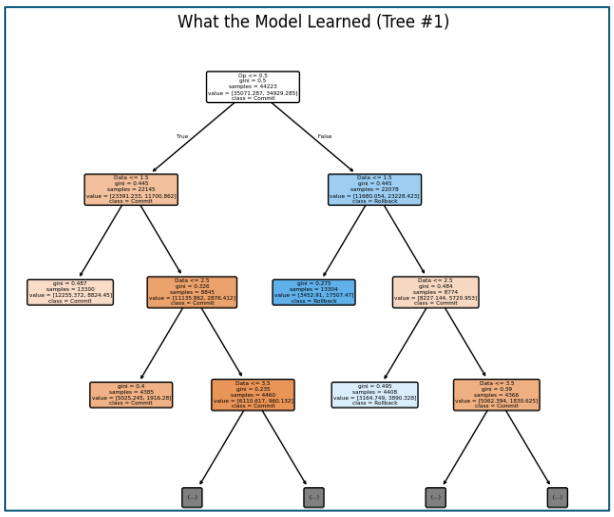

Figure 2 Random Forest Learned

4.3. Hybrid Model Performance

The Hybrid Model's performance changed significantly when a preemption threshold was added to the model. This section is devoted to the Hybrid Model. After we made this modification to the Hybrid Model, the performance of the hybrid model changed. The number of rollbacks has decreased significantly. More than 200 now. When we compare this to what we typically observe, which serves as our baseline for actual rollbacks, we see a decrease of more than 90%. We prevented approximately 3,300 transactions. Before these transactions could begin, they were stopped. They were unable to enter the system due to issues. There are 3,300 blocked transactions altogether. We took this action to ensure that the transactions would run smoothly. The 3,300 transactions were preempted.

Resource Utilisation: We can see that there are rollbacks now. This indicates that transactions that are likely to fail are not wasting system resources, such as locks, or processing time. This decrease in rollbacks results in more effective utilisation of system resources like processing time and locks.

5. DISCUSSION

5.1. Mitigation of Thrashing via Preemption

The primary finding of this study is that the "thrashing" phenomenon described in early concurrency control literature [4] is effectively mitigated by machine learning-based preemption. The 3,500 rolled-back transactions in the baseline 2PL scenario are a significant waste of system resources because they acquire locks, block other valid users, and consume CPU cycles only to be terminated later. These "expensive failures" were successfully transformed into "inexpensive preemptions" by the Hybrid model. By blocking doomed transactions at the gate, the system preserved resources for the 65% of transactions destined to commit.

5.2. Predictive Accuracy vs. Resource Utilisation

The high precision of the "Rollback" prediction class is indicated by the Hybrid model's low number of actual rollbacks (200). By ensuring that only "safe" transactions entered the critical section, the model effectively served as a filter. Although the total number of rejected transactions(Rollbacks and Preempted) remained comparable to the baseline, preempted transactions are theoretically less likely to have an effect on system latency because they do not hold locks.

5.3. Limitations and Future Work

The current findings are based on a static data skew in which Data Item 1 remains the "hot spot" indefinitely. Hot spots change over time in a dynamic production environment (such as moving from Data Item 1 to Data Item 5). Without retraining, a static Random Forest model would fail to adapt to such shifts. Implementing an Online Learning framework, in which the classifier is periodically retrained on recent transaction logs to adapt to changing contention patterns, will be the focus of future research [10].

CONCLUSION

This paper presents a hybrid concurrency control paradigm that combines Two-Phase Locking with a Random Forest classifier to minimize performance degradation in high-contention situations. After being trained on historical execution data with an ensemble of estimators and balanced class weights, the classifier makes predictions about the outcomes of transactions

based on important characteristics like the operation type and the lock requested. By substituting low-cost early aborts for costly rollbacks, experiments show that employing a pre-execution filter with a predetermined probability threshold significantly improves efficiency. The suggested model, in the end, maintains the serializability guarantees of the conventional locking protocol while simultaneously achieving faster throughput and lower lock contention.

ACKNOWLEDGEMENT

The main author, Dr. Sonal Kanungo, sincerely acknowledges the contributions of the co-authors and colleagues for their valuable support, guidance, and constructive feedback throughout this research.

REFERENCES

[1] P. A. Bernstein, V. Hadzilacos, and N. Goodman, Concurrency Control and Recovery in Database Systems. Reading, MA, USA: Addison-Wesley, 1987.

[2] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[3] G. Weikum and G. Vossen, Transactional Information Systems: Theory, Algorithms, and the Practice of Concurrency Control and Recovery. San Francisco, CA, USA: Morgan Kaufmann, 2001.

[4] R. Agrawal, M. J. Carey, and M. Livny, "Concurrency control performance modeling: Alternatives and implications," in ACM Transactions on Database Systems (TODS), vol. 12, no. 4, pp. 609–654, Dec. 1987.

[5] A. Silberschatz, H. F. Korth, and S. Sudarshan, Database System Concepts, 7th ed. New York, NY, USA: McGraw-Hill Education, 2019.

[6] A. Pavlo et al., "Self-driving database management systems," in Proc. 8th Biennial Conf. Innovative Data Syst. Res. (CIDR), Chaminade, CA, USA, Jan. 2017.

[7] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, Oct. 2011.

[8] J. Gray, P. Sundaresan, S. Englert, K. Baclawski, and P. J. Weinberger, "The dangers of replication and a solution," in Proc. ACM SIGMOD Int. Conf. Management of Data, Montreal, QC, Canada, 1996, pp. 173–182.

[9] T. Kraska, M. Alizadeh, A. Beutel, E. H. Chi, and J. Dean, "SageDB: A learned database system," in Proc. 9th Biennial Conf. Innovative Data Syst. Res. (CIDR), Polson, MT, USA, Jan. 2019.

[10] D. Van Aken, A. Pavlo, G. J. Gordon, and B. Zhang, "Automatic database management system tuning through large-scale machine learning," in Proc. ACM SIGMOD Int. Conf. Management of Data, Chicago, IL, USA, 2017, pp. 1009–1024.

[11] J. R. Quinlan, "Induction of decision trees," Machine Learning, vol. 1, no. 1, pp. 81–106, Mar. 1986.

[12] Y. C. Tay, N. Goodman, and R. Suri, "Locking performance in centralized databases," in ACM Transactions on Database Systems (TODS), vol. 10, no. 4, pp. 415–462, Dec. 1985.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, Jun. 2002.

[14] M. Stonebraker, "The case for shared nothing," IEEE Database Engineering Bulletin, vol. 9, no. 1, pp. 4–9, Mar. 1986.

[15] H. Kimura, G. Graefe, and H. Kuno, "Efficient locking for multicore architectures," in Proc. ACM SIGMOD Int. Conf. Management of Data, Athens, Greece, 2011, pp. 1243–1246.

About the Authors

Dr. Sonal Kanungo Sharma

Dr. Sonal Kanungo Sharma is a distinguished academic with a PhD in Computer Science from Veer Narmad South Gujarat University, Surat. She holds a M.Sc in Information Technology and a Bachelor’s degree in Computer Applications (BCA) from DAVV Indore. With 18 years of teaching and administration experience at prestigious institutions such as TIMSCDR, Mumbai, DPG School of Technology & Management, JIMS Vasant Kunj, et-al, in the Delhi NCR region under MDU Rohtak and GGS-IPU Delhi, in addition to a long tenure with ZSP School of Technology & Management under VNSGU Surat, etc., Dr. Sonal has made significant contributions to the field of education.

Her extensive teaching portfolio encompasses Post Graduate and Graduate courses under Computer Applications (MCA/ BCA), Management (BBA), and Engineering disciplines (B. Tech/ M. Tech), covering advanced subjects like Advanced DBMS, Object-

Oriented Programming, Web Technologies, Operating Systems, Cloud Computing, IoT, Data Warehousing, Data Science, Python, and more. Her teaching methodology integrates a defined syllabus, industry projects, research, and paper writing.

Dr. Ashwini Renavikar

Designation: Professor and Head of department for Integrated MCA program. Thakur Institute of Management Studies, Career Development and Research, Mumbai. She completed her doctorate in computer management from Pune University in 2012. She has more than 25 years of experience to her credit which includes teaching, research and industrial experience. She has authored 5 text and reference books and more than 25 research papers at the national and international levels. Dr Ashwini is also an approved PhD guide for 2 international universities. She has funded research projects for Pune University and the Ministry of HRD, Government of India.

She is a certified software quality analyst by the Quality Assurance Institute (USA) and a certified Business English Communication expert certified by Cambridge University. She has 4 patents (2 Indian and 2 international) to her credit. She is also a certified soft skill trainer and has conducted soft skill workshops for students and professionals.

She worked as a software quality analyst at Shrisoft Pvt Ltd, a company in ERP development. Also, she was associated as a visiting faculty in computer science subjects at Symbiosis University and UNCAF, a European university.

Presently pursuing a post-doctorate in the field of AI, jointly offered by Langbustech University (New Jersey) and D Y Patil University. Recently applied for a funded project for ICSSR and IKS. Co-guide to 1 student who completed a PhD in computer science in 2023.

Dr. Shiksha Dubey

Dr. Shiksha Dubey received her doctorate in machine learning from VJTI, Matunga. In 2016, she also passed the SET in Computer Application. She is well-versed in the fields of machine learning and artificial intelligence. Over the course of her roughly twelve years of teaching expertise, she instructed students in computer science and computer applications in a variety of technical subjects. Her areas of expertise include Java, RPA, digital forensics, and deep learning. She has also made a noticeable contribution to the field of research, as evidenced by the numerous research papers she has published.

Dr. Gayatri Kapadia

Dr. Gayatri Kapadia is an Assistant Professor in the Department of Computer Applications (MCA), Sarvajanik College of Engineering and Technology, Sarvajanik University. She has over 18 years of teaching experience in higher education. She holds M.C.A., M.Phil. in Computer Science, and Ph.D. degrees. Her teaching expertise includes Data Structures, Cloud Computing, NoSQL Databases, Data Management, and Requirement Engineering. She currently serves as the Head of the Department (MCA) and actively contributes to academic leadership, curriculum development, and quality assurance initiatives.