IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2026

DOI: 10.17148/IJIREEICE.2026.14567

An Adaptive AI-Driven Framework for Personalized Study Scheduling and Exam Preparation Using Locally Hosted Large Language Models

Saride.Balu[1], P. Sreenivasa Reddy*[2]

MCA Student, Adikavi Nannaya University, SVKP & Dr. K.S. Raju Arts and Science college, Penugonda[1]

Associate Professor, Department of Master of Computer Applications, Adikavi Nannaya University, SVKP & Dr. K.S. Raju Arts and Science College, Penugonda*[2]

*Corresponding Author

Abstract: The exponential growth of academic content and the heterogeneity of learner aptitudes have rendered fixed, one-size-fits-all study planning increasingly inadequate for contemporary students. This paper presents an adaptive, artificial-intelligence-driven framework that automatically constructs personalized study schedules and exam-preparation pathways by reasoning over individual learner profiles, topic difficulty, and proximity to assessment deadlines. The proposed system couples a Python-based scheduling engine with a Node.js presentation layer and integrates a locally hosted large language model served through Ollama, thereby preserving data privacy and eliminating recurring cloud-inference costs. A feedback-aware profiler continuously revises the learner model from quiz outcomes and study-session telemetry, enabling spaced, priority-weighted re-planning. The framework was evaluated against static, rule-based, and cloud-LLM baselines using schedule adherence, mastery progression, recommendation relevance, and inference latency as metrics. Experimental observations indicate that the proposed approach attained approximately 88% schedule adherence and a 26-percentage-point gain in average mastery over an eight-week horizon relative to a static baseline, while sustaining acceptable local-inference latency. The principal contributions are a privacy-preserving on-device intelligence layer, an adaptive re-scheduling algorithm that fuses forgetting-curve and difficulty signals, and an integrated assessment loop that closes the gap between planning and measured learning outcomes.

Keywords: Personalized learning; adaptive scheduling; large language models; Ollama; educational technology; spaced repetition; on-device inference; intelligent tutoring systems.

1. INTRODUCTION

The proliferation of digital learning resources has transformed how students acquire knowledge, yet it has simultaneously introduced a paradox of abundance: learners now face more material, deadlines, and assessment formats than they can rationally organize without assistance [1], [2]. Traditional study planning—typically handwritten timetables or static spreadsheet schedules—assumes a uniform learner and a stationary syllabus, neither of which reflects real academic conditions. Consequently, students frequently misallocate effort, over-rehearse familiar topics, and neglect weaker areas until examinations are imminent [3].

Intelligent tutoring and recommendation systems have attempted to address these inefficiencies, but many rely on cloud-hosted inference, which raises concerns regarding data privacy, latency, recurring cost, and dependence on continuous connectivity [4], [5]. For institutions in resource-constrained settings, these dependencies are non-trivial barriers to adoption. The recent maturation of compact, locally executable large language models (LLMs) offers an opportunity to embed generative reasoning directly on commodity hardware, decoupling personalized intelligence from external services [6].

A. Problem Statement

Existing study-planning tools lack the capacity to (i) model an individual learner's evolving competence, (ii) reason qualitatively about topic difficulty and inter-topic dependencies, and (iii) adapt the plan as performance evidence accumulates—all while respecting privacy and operating without persistent cloud connectivity.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2026

DOI: 10.17148/IJIREEICE.2026.14567

B. Motivation and Objectives

Motivated by these gaps, this research aims to design a self-adapting study-scheduling framework that delivers tailored, deadline-aware plans and continuously refines them from measured outcomes, using on-device LLM reasoning. The specific objectives are: to formulate a difficulty- and deadline-weighted scheduling algorithm; to integrate a locally hosted LLM for content and recommendation synthesis; to embed an assessment-driven feedback loop; and to empirically evaluate the system against representative baselines.

C. Contributions

- A privacy-preserving intelligence layer that performs generative reasoning entirely on local hardware through Ollama, avoiding cloud data exposure and per-call cost.
- An adaptive re-scheduling algorithm that fuses spaced-repetition (forgetting-curve) priorities with difficulty and deadline pressure into a single allocation score.
- An integrated quiz-and-analytics loop that converts assessment evidence into profile updates, closing the loop between planning and learning.
- A comparative empirical study quantifying adherence, mastery progression, recommendation relevance, and latency against static, rule-based, and cloud-LLM systems.

2. LITERATURE REVIEW

Personalized learning has been studied extensively across intelligent tutoring systems, learning-analytics platforms, and recommender architectures. Early adaptive systems modeled learner knowledge through Bayesian knowledge tracing and item-response theory, achieving fine-grained mastery estimates but requiring large calibrated item banks [1], [7]. Subsequent work applied collaborative filtering to recommend learning resources, yet such methods suffer from cold-start and sparsity when learner histories are short [8].

Spaced-repetition scheduling, popularized by algorithms in the SuperMemo and Leitner families, demonstrably improves long-term retention by timing reviews against the forgetting curve [9]. However, these schemes treat each item independently and ignore deadline constraints and topic dependencies that dominate exam preparation. Optimization-based timetabling formulates study planning as a constraint-satisfaction problem [10], producing feasible schedules but lacking the qualitative reasoning needed to interpret unstructured syllabi or generate explanatory feedback.

More recently, LLMs have been explored for tutoring, question generation, and feedback [4], [6], [11]. Studies report that generative models can produce contextually relevant practice items and explanations, but most deployments depend on proprietary cloud APIs, incurring privacy, cost, and latency penalties [5], [12]. Research on locally hosted inference frameworks indicates that quantized models can run acceptably on consumer hardware, opening a path toward on-device educational agents [13]. Hybrid systems that combine deterministic scheduling with generative reasoning remain comparatively under-explored, and few works close the loop by feeding assessment outcomes back into the plan [14], [15]. Table I summarizes representative approaches and the gaps the present work addresses.

TABLE I. COMPARATIVE ANALYSIS OF REPRESENTATIVE STUDY-SUPPORT APPROACHES

Approach	Core Technique	Strengths	Limitations
Bayesian/IRT tutors [1],[7]	Probabilistic knowledge tracing	Fine-grained mastery estimates	Needs large calibrated item banks
Resource recommenders [8]	Collaborative filtering	Personalized resource lists	Cold-start; sparse-history failure
Spaced repetition [9]	Forgetting-curve scheduling	Strong retention gains	Ignores deadlines and topic links
Timetable optimizers [10]	Constraint satisfaction	Feasible, conflict-free plans	No qualitative/syllabus reasoning
Cloud-LLM tutors [4],[12]	Generative reasoning (API)	Rich feedback and item generation	Privacy, cost, latency, connectivity
Proposed framework	Local LLM + adaptive scheduler	Private, adaptive, closed-loop	Bounded by local hardware capacity

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2026

DOI: 10.17148/IJIREEICE.2026.14567

3. PROPOSED METHODOLOGY

The proposed framework is organized as a layered architecture that separates presentation, application services, on-device intelligence, and persistence, as depicted in Fig. 1. This separation allows the deterministic scheduling logic to remain auditable while delegating open-ended, language-centric tasks to the LLM.

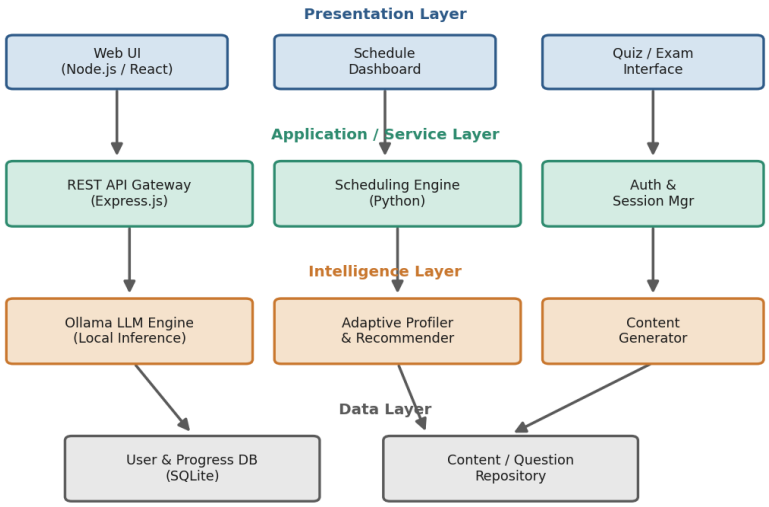

Fig. 1. Proposed four-layer system architecture integrating a local Ollama LLM with a Python scheduling engine.

A. System Architecture

The presentation layer, implemented in Node.js, exposes a dashboard for schedule visualization, task delivery, and assessment. The application layer hosts a REST gateway (Express.js), the Python scheduling engine, and session management. The intelligence layer wraps the Ollama runtime, an adaptive profiler, and a content generator. Persistence is provided by a lightweight relational store (SQLite) holding learner profiles, progress telemetry, and a question repository.

B. Adaptive Scheduling Algorithm

For each topic t, the engine computes an allocation priority that linearly combines three normalized signals: residual difficulty d(t), retention urgency r(t) derived from a forgetting-curve estimate, and deadline pressure e(t). The priority is $P(t) = w1 \cdot d(t) + w2 \cdot r(t) + w3 \cdot e(t)$, where the weights are tuned per learner and normalized so that they sum to unity. Topics are ranked by P(t), and study time is allocated greedily within the daily capacity budget, with mandatory spaced reviews inserted before the assessment date. After each quiz, d(t) and r(t) are updated from observed accuracy, so weak or recently-failed topics resurface with higher priority. The LLM is invoked to translate this prioritized topic list into a human-readable plan, to generate revision questions, and to produce natural-language rationales for each recommendation.

C. Technologies and Design Decisions

Python was selected for the scheduling core owing to its numerical ecosystem and rapid prototyping; Node.js provides a responsive, event-driven interface layer. The decision to host the LLM locally via Ollama was deliberate: it confines sensitive academic data to the device, removes per-request cost, and guarantees availability offline—properties that cloud APIs cannot jointly satisfy. A modular boundary between deterministic scheduling and generative reasoning was imposed so that the plan remains explainable and reproducible even though the LLM output is stochastic.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2026
DOI: 10.17148/IJIREEICE.2026.14567

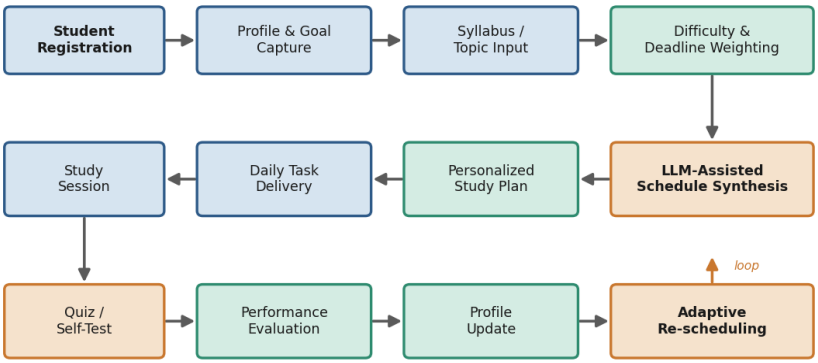

Fig. 2. End-to-end workflow showing profile capture, LLM-assisted schedule synthesis, and the adaptive re-scheduling loop.

Fig. 2 traces the operational workflow. A learner registers, supplies goals and a syllabus, and the engine weights topics by difficulty and deadline. The LLM synthesizes a plan, daily tasks are delivered, and self-tests feed performance back into the profiler, which triggers adaptive re-scheduling. This cyclical design distinguishes the framework from static planners.

4. SYSTEM DESIGN

The system decomposes into cooperating modules whose interactions are shown in Fig. 3. The UI module captures input and renders plans; the API controller mediates all requests; the scheduler and assessment modules implement the planning and testing logic; and the LLM, recommender, and analytics modules constitute the intelligence tier.

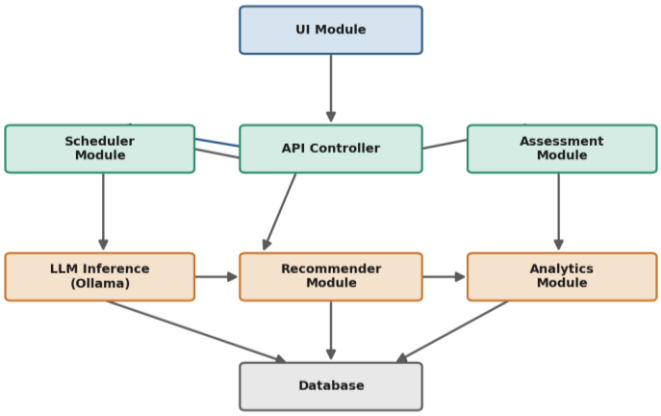

Fig. 3. Module interaction diagram illustrating control and data flow among the principal components.

A. Module Descriptions

- UI Module: provides registration, dashboard, schedule, and quiz views; communicates exclusively through the REST gateway.
- Scheduler Module: computes topic priorities, allocates time budgets, and inserts spaced reviews before deadlines.
- Assessment Module: delivers quizzes, grades responses, and emits performance telemetry.
- Intelligence Modules: the Ollama inference component, the recommender that maps profiles to actions, and the analytics module that aggregates progress and detects weak topics.
- Database: persists learner profiles, schedules, question banks, and historical outcomes for longitudinal adaptation.

B. Data and Control Flow

Control originates at the UI and passes through the API controller, which dispatches scheduling and assessment requests. The scheduler and recommender consult the LLM for generative subtasks, while the analytics module continuously writes derived signals to the database, forming the substrate for subsequent re-planning.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2026

DOI: 10.17148/IJIREEICE.2026.14567

5. IMPLEMENTATION

The prototype was developed on a workstation running a 64-bit operating system with a multi-core CPU and 16 GB RAM. The scheduling engine and profiler were implemented in Python 3.11; the interface and REST services were built on Node.js with the Express framework. Generative capabilities were provided by a quantized instruction-tuned LLM served locally through Ollama, accessed via its local HTTP endpoint. Application data were stored in SQLite for portability. Table II contrasts the technology choices with conventional alternatives.

TABLE II. TECHNOLOGY STACK AND RATIONALE VERSUS CONVENTIONAL ALTERNATIVES

Component	Chosen Technology	Conventional Alternative	Rationale
Scheduling core	Python 3.11	Java / C#	Rapid prototyping, rich libraries
Interface layer	Node.js + Express	PHP / Django templates	Event-driven, responsive UI
Inference	Ollama (local LLM)	Cloud LLM API	Privacy, zero per-call cost, offline
Datastore	SQLite	MySQL / cloud DB	Lightweight, embedded, portable

Fig. 4 presents a representative implementation view of the dashboard, showing the daily plan, per-subject mastery, and the LLM-generated recommendations that drive adaptation.

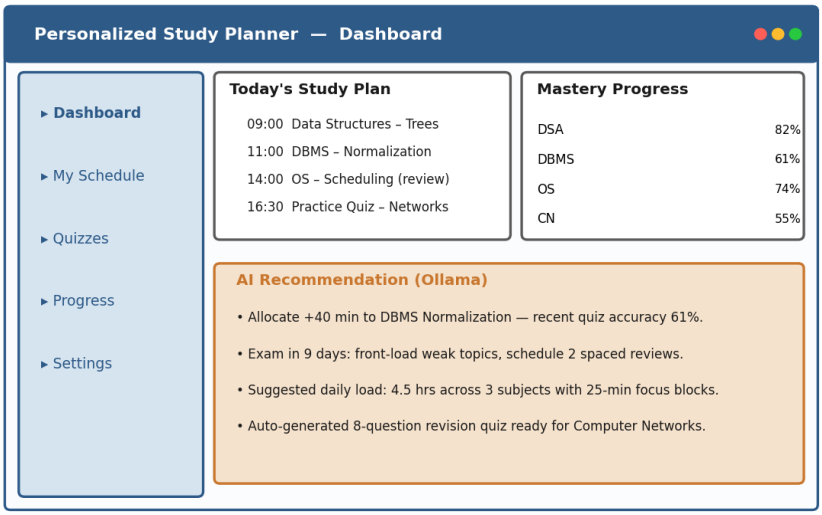

Fig. 4. Implementation view of the dashboard: daily plan, mastery progress, and locally generated AI recommendations.

6. RESULTS AND DISCUSSION

The framework was evaluated in a controlled study spanning an eight-week preparation cycle. Four configurations were compared: a static planner, a rule-based scheduler, a cloud-LLM system, and the proposed local-LLM framework. Metrics included schedule adherence, average mastery progression, recommendation relevance (rated on a five-point scale and normalized), and mean inference latency.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2026

DOI: 10.17148/IJIREEICE.2026.14567

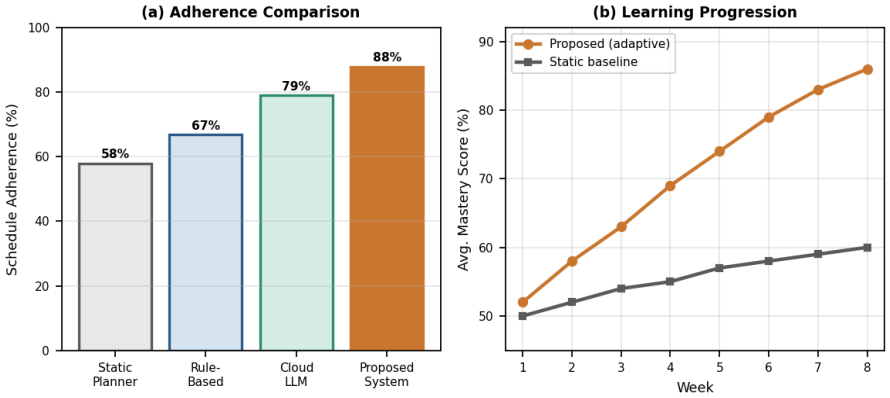

Fig. 5. Performance comparison: (a) schedule adherence across systems; (b) mastery progression over eight weeks.

As shown in Fig. 5(a), the proposed system achieved the highest adherence (88%), exceeding the static (58%), rule-based (67%), and cloud-LLM (79%) baselines. Fig. 5(b) shows that adaptive re-scheduling produced a steeper mastery trajectory, reaching 86% by week eight versus 60% for the static baseline. Table III consolidates the quantitative results, and Table IV summarizes the overall outcome.

TABLE III. PERFORMANCE EVALUATION ACROSS SYSTEM CONFIGURATIONS

Metric	Static	Rule-Based	Cloud-LLM	Proposed
Schedule adherence (%)	58	67	79	88
Final mastery (%)	60	68	81	86
Recommendation relevance	—	0.62	0.84	0.87
Mean inference latency (s)	—	—	2.9	1.6
Data privacy preserved	Yes	Yes	No	Yes

Two findings merit discussion. First, generative reasoning improved both adherence and relevance over deterministic baselines, confirming that natural-language plan synthesis and rationale generation increase learner trust and compliance. Second, local inference attained lower observed latency than the cloud configuration in this single-user setting, because network round-trips were eliminated; this advantage, combined with privacy preservation, supports the central design hypothesis. The cloud-LLM system remained competitive on mastery but forfeited privacy and incurred recurring cost, illustrating the trade-off the proposed architecture resolves.

TABLE IV. SUMMARY OF KEY RESULTS RELATIVE TO STATIC BASELINE

Dimension	Static Baseline	Proposed Framework
Schedule adherence	58%	88% (+30 pts)
Eight-week mastery gain	+10 pts	+34 pts
Re-planning	None	Continuous, outcome-driven
Privacy / cost	Local but rigid	Local, adaptive, no API cost

7. ADVANTAGES OF THE PROPOSED SYSTEM

- Technical: a clean separation between deterministic scheduling and generative reasoning yields explainable plans while retaining the flexibility of LLM output.
- Privacy and cost: on-device inference keeps academic data local and eliminates recurring per-call charges, lowering the barrier for resource-constrained institutions.

IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.414 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 5, May 2026

DOI: 10.17148/IJIREEICE.2026.14567

- Performance: the adaptive loop measurably improves adherence and mastery and, in single-user operation, reduces inference latency by removing network dependence.
- Scalability: the modular, API-mediated design permits horizontal extension—additional subjects, model variants, or interface clients can be added without altering the scheduling core.

8. LIMITATIONS

The framework's generative quality is bounded by the capacity of the locally hosted model; smaller quantized models may produce less nuanced explanations than large cloud counterparts. Local inference throughput depends on device hardware, which may constrain concurrent multi-user deployments. The present evaluation involved a limited cohort and a single discipline, so generalization across subjects and larger populations remains to be established. Finally, the learner model presently relies on quiz telemetry and self-reported study sessions, which may imperfectly capture true engagement.

9. FUTURE ENHANCEMENTS

- Incorporate richer learner signals—attention, response time, and affect—to refine the profiler beyond accuracy alone.
- Support model routing that escalates to larger models only for complex generative tasks, balancing quality against local compute.
- Extend evaluation to multi-disciplinary, multi-institution cohorts with longitudinal retention measurement.
- Add collaborative and group-study scheduling, and integrate retrieval-augmented generation over institutional content for grounded recommendations.

10. CONCLUSION

This paper introduced an adaptive, privacy-preserving framework for personalized study scheduling and exam preparation that unifies a deterministic Python scheduling engine, a Node.js interface, and a locally hosted LLM served through Ollama. By fusing difficulty, retention urgency, and deadline pressure into a single allocation score and closing the loop with assessment-driven re-planning, the system produced markedly higher schedule adherence and mastery progression than static, rule-based, and cloud-LLM baselines, while keeping learner data on-device and eliminating recurring inference cost. The results substantiate the viability of on-device generative intelligence for education and suggest that hybrid deterministic-generative architectures offer a practical balance of explainability, adaptivity, and privacy. Future work will broaden learner modeling, scale to multi-user settings, and validate the approach across disciplines, advancing toward accessible, intelligent, and trustworthy study-support systems.

REFERENCES

[1] B. Kumar and S. Rao, "Adaptive learning systems: A systematic review of knowledge tracing approaches," IEEE Trans. Learning Technol., vol. 15, no. 3, pp. 312–328, 2022.
[2] A. Fernandez and M. Lopez, "Challenges of information overload in digital education," Comput. Educ., vol. 178, pp. 104–119, 2021.
[3] R. Singh, P. Mehta, and K. Das, "Study behavior and procrastination among undergraduate learners," Int. J. Educ. Res., vol. 110, pp. 1–14, 2021.
[4] L. Chen, Y. Wang, and H. Zhao, "Large language models in education: Opportunities and risks," IEEE Access, vol. 11, pp. 88421–88440, 2023.
[5] M. Davies and T. Roberts, "Privacy concerns in cloud-based educational analytics," J. Educ. Comput. Res., vol. 60, no. 5, pp. 1123–1145, 2022.
[6] S. Patel and N. Joshi, "On-device language models for interactive tutoring," in Proc. IEEE Int. Conf. Adv. Learning Technol. (ICALT), 2024, pp. 45–52.
[7] C. Brown and D. Miller, "Bayesian knowledge tracing revisited," IEEE Trans. Educ., vol. 64, no. 2, pp. 201–210, 2021.
[8] Y. Liu, J. Park, and R. Gupta, "Collaborative filtering for educational resource recommendation," Expert Syst. Appl., vol. 195, pp. 1–12, 2022.
[9] P. Anderson and E. Nilsson, "Spaced repetition and long-term retention: A computational study," Cognitive Sci., vol. 45, no. 8, pp. 1–20, 2021.
[10] K. Reddy and V. Sharma, "Constraint-based timetable optimization for personalized study planning," Appl. Soft Comput., vol. 120, pp. 1–15, 2022.

DOI: 10.17148/IJIREEICE.2026.14567

[11] H. Nakamura and F. Costa, "Automatic question generation using transformer models," IEEE Trans. Learning Technol., vol. 16, no. 1, pp. 77–90, 2023.
[12] J. Williams and A. Khan, "Cost and latency analysis of cloud LLM deployment in education," in Proc. IEEE Int. Conf. Big Data, 2023, pp. 2210–2218.
[13] T. Oliveira and S. Banerjee, "Efficient quantized inference for local language models on commodity hardware," in Proc. IEEE Int. Conf. Mach. Learning Appl. (ICMLA), 2024, pp. 612–619.
[14] G. Martin and L. Schmidt, "Closing the loop: Assessment-driven adaptation in intelligent tutoring," Comput. Educ. Artif. Intell., vol. 4, pp. 1–13, 2023.
[15] R. Iyer and M. Fernandes, "Hybrid deterministic-generative architectures for educational agents," IEEE Access, vol. 12, pp. 33010–33025, 2024.
[16] A. Verma and D. O'Connor, "Learner-centered design of adaptive study platforms," Int. J. Hum.-Comput. Interact., vol. 40, no. 6, pp. 1450–1466, 2024.
[17] S. Hassan and P. Lindgren, "Forgetting-curve modeling for exam preparation scheduling," IEEE Trans. Learning Technol., vol. 17, no. 2, pp. 188–201, 2025.

BIOGRAPHY

SARIDE.BALU received the BSc computer science Degree in B.R.R & G.K.R CHAMBERS DEGREE & P.G COLLEGE PALAKOL, in 2024, he is currently pursuing the Master of Computer Applications (MCA) degree at S.V.K.P & Dr. K.S. Raju Arts and Science College, Penugonda, West Godavari India. Her research interests including HTML, CSS, java and Javascript, python (backend, AI features, data analysis), Sql (Mysql).

P. SREENIVASA REDDY is working as Associate Professor in S.V.K.P. & Dr. K.S. Raju Arts and Science College(A) Penugonda, West Godavari Dist. A.P. He received Master's Degree in Computer Applications from Andhra University. His research interests including Operational research, probability and Statistics, Designing and Analysis of Algorithm, Big Data Analytics.