iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: https://doi.org/10.22214/ijraset.2026.85563

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue X Oct 2026- Available at www.ijraset.com

QGenAI: A Web-Based System for Competitive Examination Practice, Difficulty-Stratified Analytics, and Mistake-Driven AI Tutoring

Abishek M P[1], Dinesh E[2], Gowtham S[3], Mohana Preetha S[4]
Department of CSE, Adhiyamaan College Of Engineering, Hosur, India

Abstract: ***Preparing for national competitive examinations such as the Joint Entrance Examination (JEE), the National Eligibility cum Entrance Test (NEET), and the Graduate Aptitude Test in Engineering (GATE) requires regular multiple-choice question (MCQ) practice, reliable answer verification, and systematic mistake revision. When aspirants rely on disconnected textbooks, static question portals, and standalone chat tools, practice records become fragmented and conversational assistance lacks awareness of a learner's actual errors. This paper presents QGenAI, an authenticated web-based practice, difficulty analysis, and personalized tutoring platform built for JEE, NEET, and GATE preparation. Learners configure custom practice sessions by selecting an examination profile, subject, topic, difficulty tier (Easy, Medium, or Hard), and question count (1, 5, 10, 15, or 20). The application retrieves active MCQs from a structured question bank while withholding answer keys from client payloads to preserve assessment integrity. Submitted responses are graded server-side, persisted with idempotent retry logic, and aggregated into account-scoped analytics covering session completion, difficulty-wise accuracy, seven-day activity, and topic-level mistake queues. Furthermore, an integrated AI Tutor module connects to external large language model (LLM) providers (Groq, Gemini, OpenAI, or Mistral) to deliver both general topic lessons and targeted remediation grounded in up to five of the learner's most recent incorrect attempts. The system is implemented using React, Vite, Python FastAPI, and Supabase PostgreSQL with Row-Level Security (RLS). We detail the four-tier architecture, relational schema, REST interface, evaluation formulation, and a worked empirical example across 28 completed sessions.***

Keywords: competitive examination preparation, multiple-choice questions, learning analytics, intelligent tutoring systems, large language models, FastAPI, React, Supabase, Row-Level Security.

I. INTRODUCTION

National competitive entrance examinations in India—specifically JEE for undergraduate engineering, NEET for medical studies, and GATE for postgraduate engineering and public-sector recruitment—test thousands of syllabus concepts under strict time and accuracy constraints. High-scoring preparation depends on three tightly coupled activities: solving examination-specific multiple-choice questions (MCQs) at graduated difficulty levels, reviewing explanations for incorrect attempts, and monitoring longitudinal accuracy across subjects and topics. In conventional workflows, students switch between printed question banks, online test series, manual spreadsheets, and general-purpose conversational artificial intelligence (AI) tools. Because these tools do not share state, students must manually track weak topics, and external AI chatbots cannot automatically inspect the exact distractors a learner selected during practice. This paper describes QGenAI (developed under the full project title QGenAI: Intelligent Competitive Exam Question Generation, Difficulty Analysis and Personalized Practice System), a unified web application designed to close the loop between structured MCQ practice, difficulty-stratified performance analytics, and mistake-aware AI tutoring. While the broader architectural vision encompasses automated question synthesis, the verified production prototype intentionally retrieves vetted questions from a curated relational question bank through the /api/v1/practice/generate endpoint so that every practice item and answer key remains syllabus-accurate. The main contributions of this work are:

1) *Protected, difficulty-stratified MCQ delivery and grading:* A FastAPI service resolves examination metadata across JEE, NEET, and GATE, delivers active questions at Easy, Medium, and Hard tiers without exposing answer keys to the browser, and grades submissions server-side against trusted option records.
2) *Account-scoped learning analytics with retry-safe persistence:* Submitted attempts are recorded in Supabase PostgreSQL using conflict-safe upserts with transient network retry handling and Row-Level Security (RLS), powering a dashboard that computes overall accuracy, difficulty breakdowns, and seven-day activity.

©IJRASET: All Rights are Reserved

3) *Mistake-driven AI Tutor integration:* An optional tutoring service interfaces with configurable LLM providers (Groq, Gemini, OpenAI, or Mistral) to provide topic explanations and targeted remediation using up to five of the learner's most recent incorrect MCQ attempts per topic without exposing personally identifiable information.

The remainder of the paper is organized as follows. Section II reviews related literature, Section III defines the problem statement and objectives, Section IV presents the four-tier system architecture and database schema, Section V details the methodology, scoring rules, and REST endpoints, Section VI demonstrates the implementation with a worked evaluation across 28 sessions, Section VII discusses current limitations and future extensions, and Section VIII concludes the paper.

II. LITERATURE REVIEW

Practice testing and retrieval practice are among the most thoroughly validated techniques in cognitive and educational psychology. Meta-analyses by Adesope et al. [15] across practice-testing formats and by Yang et al. [11] across 222 classroom studies demonstrate that frequent quizzing combined with corrective feedback significantly improves conceptual retention and transfer compared with passive restudying.

In higher education, systematic reviews of learning analytics by Viberg et al. [14] and Ifenthaler and Yau [12], alongside predictive performance models by Akçapınar et al. [13], Riestra-González et al. [9], Shou et al. [5], and Wang et al. [2], confirm that fine-grained session logs help identify at-risk topics early—provided that learner privacy and data governance principles are enforced through strict access controls.

More recently, large language models have reshaped virtual tutoring. Surveys and reviews by Wollny et al. [10], Kasneci et al. [8], M. Al-Emran and Malik [3], and Egara and Mosia [1] outline how conversational agents can deliver on-demand explanations, while empirical evaluations by Pardos and Bhandari [7] and Daheim et al. [6] show that LLM-generated hints and stepwise error remediation can match human tutor effectiveness when properly grounded. However, a field experiment by Bastani et al. [4] warns that unrestricted generative AI during practice can act as a crutch and harm subsequent unassisted exam performance unless pedagogical guardrails are applied. QGenAI directly operationalizes this finding: learners complete unassisted MCQ sessions graded strictly against a verified database first, and the LLM tutor is invoked only after submission to remediate recorded mistakes.

III. PROBLEM STATEMENT AND OBJECTIVES

Given a multi-exam question bank covering JEE, NEET, and GATE across multiple subjects, topics, and difficulty levels, the platform must: (a) authenticate learners and isolate their personal practice records, (b) deliver configurable MCQ sessions without leaking correct answers to the client prior to submission, (c) evaluate answers reliably under transient network conditions and summarize performance across difficulty levels and topics, and (d) supply context-aware AI tutoring anchored to the learner's actual completed mistakes.

The primary objectives are to eliminate manual progress tracking, prevent answer-key inspection in browser network payloads, and bridge the gap between quantitative practice analytics and qualitative conceptual remediation using commodity open-source web technologies.

IV. SYSTEM ARCHITECTURE

A. Overview

QGenAI follows a modular four-tier architecture illustrated in Fig. 1. The Presentation Tier is a responsive React single-page application bundled with Vite that renders the authentication screens, session builder, MCQ practice workspace, analytics dashboard, and AI Tutor interface.

The Application Tier is a Python FastAPI service mounted under /api/v1 that enforces bearer-token authentication, validates master-data hierarchies, serves sanitized question payloads, grades submissions, and orchestrates tutor prompts. The Data Tier relies on Supabase Authentication and a managed PostgreSQL database protected by Row-Level Security (RLS) policies. Finally, the External LLM Tier connects asynchronously over HTTPS via httpx to a backend-configured provider (Groq, Google Gemini, OpenAI, or Mistral) to generate topic lessons and mistake-review explanations.

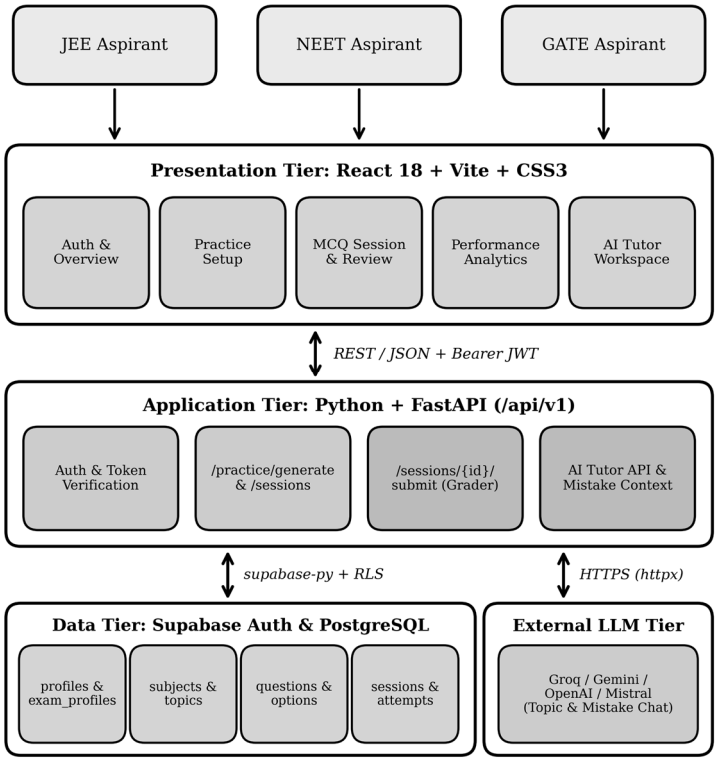


Fig. 1 Four-tier system architecture of the QGenAI platform

B. Technology Stack

Table I summarizes the software stack across all system layers. Separating the React frontend from the FastAPI microservice ensures that sensitive operations—such as answer-key lookup and LLM API key management—remain strictly server-side.

TABLE I
TECHNOLOGY STACK

Layer	Technology
Front end	React 18, Vite, JavaScript (ES6+), HTML5, CSS3, Supabase JS Client
Back end	Python 3, FastAPI, Uvicorn (ASGI), Pydantic, HTTPX, python-dotenv
Database & Auth	Supabase PostgreSQL, Supabase Auth (JWT), Row-Level Security (RLS)
AI Tutor Providers	Configurable REST integration for Groq, Gemini, OpenAI, or Mistral

C. Database Design

The PostgreSQL schema comprises ten principal relational tables summarized in Table II. Master catalog tables (exam_profiles, exam_subjects, topics, exam_difficulty_levels, and exam_question_types) enforce referential integrity so that every question in questions belongs to a valid exam-subject-topic-difficulty tuple. Each MCQ in questions links to one or more choices in question_options, where the boolean flag is_correct identifies the single valid answer. Transactional activity is captured in practice_sessions and practice_attempts, with a unique composite constraint on (practice_session_id, question_id) to prevent duplicate attempt insertion.

TABLE II
DATABASE SCHEMA AND PRINCIPAL TABLES

Table Name	Primary Purpose
profiles	Stores learner profile data linked to auth.users
exam_profiles	Defines supported competitive exams (JEE, NEET, GATE)
exam_subjects	Maps academic subjects to each examination profile
topics	Stores syllabus topics under each examination subject
exam_difficulty_levels	Defines EASY, MEDIUM, and HARD difficulty codes
exam_question_types	Defines supported item formats (currently restricted to MCQ)
questions	Stores question stem, explanation, metadata foreign keys, and active flag
question_options	Stores option text, display order, and is_correct boolean
practice_sessions	Tracks user sessions, requested parameters, count, and status
practice_attempts	Records selected answer text, is_correct outcome, and time taken

D. Access Control and Application Security

Access is governed by three defense-in-depth mechanisms summarized in Table III. First, protected FastAPI endpoints extract the JSON Web Token (JWT) from the Authorization: Bearer <token> header and verify it via supabase.auth.get_user(). Second, session and attempt writes execute through a user-scoped Supabase client (get_user_supabase) so PostgreSQL RLS policies restrict queries strictly to rows matching auth.uid() = user_id. Third, during question delivery, the backend selects only id, question_text, option_order, and option_text, stripping is_correct, answer_key, and explanation from the HTTP response until the session is formally submitted.

TABLE III
ACCESS LEVELS AND SECURITY CONTROLS

Context / Role	Permitted Functions and Security Enforcement
Unauthenticated Visitor	View landing page; register, sign in, or request password reset via Supabase Auth
Authenticated Learner	Configure JEE/NEET/GATE sessions, attempt MCQs, submit answers, view personal analytics, query AI Tutor
FastAPI Trusted Grader	Validate master metadata, strip answer keys before delivery, verify single-correct MCQ invariant, grade submissions
Database RLS Engine	Enforce user_id ownership on practice_sessions and practice_attempts; block cross-user data reads

V. METHODOLOGY

A. Authentication and Session Workflow

When a learner registers or signs in, Supabase Auth issues a signed JWT access token. The React frontend attaches this token to all API requests. Before any practice session begins, the learner selects five configuration parameters: examination code $E \in \{JEE, NEET, GATE\}$, subject S, topic T, difficulty level $D \in \{Easy, Medium, Hard\}$, and question count $N \in \{1, 5, 10, 15, 20\}$.

B. Practice Configuration and Question Retrieval

Upon receiving a configuration request, the backend resolves the corresponding foreign-key identifiers across exam_profiles, exam_subjects, topics, exam_question_types (MCQ), and exam_difficulty_levels. If any parameter is inactive or mismatched, an HTTP 400 error is returned immediately. Next, the service queries up to N active questions matching the resolved identifiers, sorts each question's options ascending by option_order, and verifies that at least N valid items exist in the bank. Simultaneously, POST /api/v1/practice/sessions inserts a new row into practice_sessions with session_status = "in_progress".

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

C. Answer Evaluation and Mistake-Driven Analytics

When the learner submits answers for a session of expected size N, the backend validates that the submitted question identifiers contain no duplicates, do not exceed N, and belong to the active examination profile. For each question $i \in \{1, ..., N\}$, the backend inspects question_options to verify that exactly one option has is_correct = True, denoted c_i, and compares it against the learner's trimmed selection a_i. The binary correctness indicator r_i is 1 if a_i matches c_i, and 0 otherwise. The session score and percentage are computed and returned.

1) *Retry-safe persistence:* Network drops between the application server and managed cloud databases can cause lost submissions. QGenAI implements upsert_practice_attempts_with_retry, which upserts evaluated attempt rows on the conflict target (practice_session_id, question_id) for up to 3 attempts with backoff delays of 0.1s and 0.25s whenever transient socket or HTTPX errors are detected. Once saved, the session status transitions from in_progress to completed.
2) *Longitudinal analytics and mistake queue:* On the client, fetchAllPages retrieves completed sessions and recorded attempts in cursor-paginated batches of 500 rows. The platform computes a topic miss rate based on incorrect and skipped attempts over total attempts. Topics with fewer than 3 attempts are flagged with an Early signal badge so learners do not over-interpret single-question variance.
3) *Privacy-preserving AI Tutor context:* When a learner clicks Teach me this topic on a queued weak topic, the backend packages up to five of the learner's most recent completed mistake records (question stem, selected option, correct option, and stored explanation) and dispatches them to the configured LLM provider. Learner names and emails are explicitly excluded.

D. REST Interface

Table IV lists the principal REST endpoints exposed by the FastAPI backend under /api/v1.

TABLE IV
MAIN REST ENDPOINTS

Endpoint	Purpose
POST /api/v1/practice/generate	Validate configuration and retrieve sanitized active MCQs
POST /api/v1/practice/sessions	Create an authenticated in_progress practice session record
POST /api/v1/practice/sessions/{id}/submit	Grade answers, persist attempts with retry, and return reviews
POST /api/v1/tutor/topic	Generate a structured AI Tutor lesson for an academic topic
POST /api/v1/tutor/mistakes	Start a remediation chat grounded in up to 5 recent topic mistakes
POST /api/v1/tutor/chat	Continue multi-turn follow-up conversation with the AI Tutor

VI. IMPLEMENTATION AND ILLUSTRATIVE EXAMPLE

A. User Interface

The React interface is organized into four primary workspace views accessible from a persistent sidebar: Overview, Practice, AI Tutor, and Analytics. The Overview screen welcomes the signed-in learner and provides quick-start navigation. The Practice builder guides the learner through a six-step configuration flow before launching the interactive test view. The Analytics screen displays top-level KPI cards, a seven-day activity bar chart, horizontal progress bars for difficulty-level accuracy, and a tabular log of recent completed sessions. The AI Tutor screen displays mistake-summary counters, a free-form prompt box, and review cards with one-click Teach me this topic actions.

B. Worked Example

To illustrate the end-to-end evaluation and analytics pipeline, we examine an actual logged user workspace comprising 28 completed MCQ sessions and 28 total question records (27 answered attempts and 1 skipped attempt) across GATE, JEE, and NEET. Across the 27 answered items, 12 were graded correct and 15 incorrect, yielding an overall answer accuracy of 44.4%. Stratified by difficulty level, the learner recorded 11/25 correct at Easy (44.0%), 0/1 at Medium (0.0%), and 1/1 at Hard (100.0%). Table V and Fig. 2 break down the 28 recorded attempts across the five practiced syllabus topics (T1 to T5) and show how the platform ranks topics for AI Tutor remediation.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

TABLE V
WORKED EVALUATION AND MISTAKE REVIEW QUEUE ACROSS FIVE TOPICS

ID	Exam & Subject	Attempts (C/I/S)	Missed (%)	Tutor Queue Status
T1	GATE CS (Prog. & DS)	10 (8 / 1 / 1)	20%	High mastery; 1 mistake queued
T2	JEE Physics (Kinematics)	5 (2 / 3 / 0)	60%	Priority remediation (3 mistakes sent)
T3	NEET Biology (Biomolecules)	11 (2 / 9 / 0)	82%	Critical weak topic (top 5 mistakes sent)
T4	NEET Biology (Plant Phys.)	1 (0 / 1 / 0)	100%	Flagged Early signal (1 attempt)
T5	NEET Biology (Genetics)	1 (0 / 1 / 0)	100%	Flagged Early signal (1 attempt)

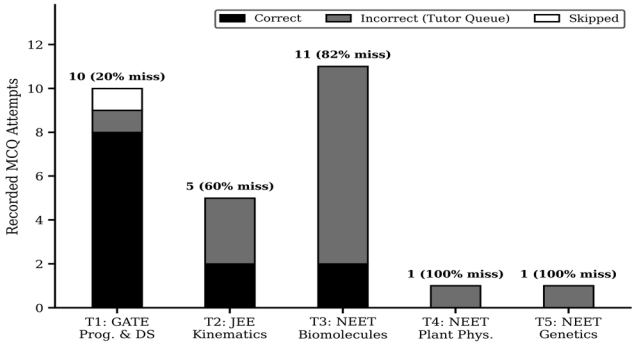


Fig. 2 Topic-wise distribution of correct, incorrect, and skipped MCQ attempts and resulting miss percentages across the five evaluated topics

As shown in Table V and Fig. 2, T1 (Programming and Data Structures) exhibits strong mastery, whereas T3 (Biomolecules) has 11 attempts with 9 incorrect responses (82% missed). Although T4 and T5 show a 100% miss rate, each rests on a single attempt; the UI therefore attaches an Early signal badge so the student prioritizes multi-attempt weak areas (T3 and T2). When the student triggers Teach me this topic on T3, the backend selects the 5 most recent incorrect Biomolecules attempts and sends their stems, chosen distractors, and stored explanations to the Groq LLM service to generate a targeted step-by-step remediation lesson.

VII. LIMITATIONS AND FUTURE WORK

While QGenAI provides a complete, secure workflow for MCQ practice, analytics, and mistake-driven tutoring, the current prototype has practical limitations that define our future work:

1) *Live AI question synthesis and validation:* The current backend retrieves verified questions from the PostgreSQL bank rather than synthesizing new items via an LLM. Future work will introduce retrieval-augmented generation (RAG) over approved syllabus documents paired with automated distractor-quality validation.
2) *Question formats and negative marking:* Competitive examinations also include Numerical Answer Type (NAT) and Multiple Select Question (MSQ) formats alongside differential negative marking. Extending the grading engine will improve exam realism.
3) *Adaptive difficulty progression:* Difficulty selection is currently manual. Incorporating Item Response Theory (IRT) or Elo-based skill ratings will allow the platform to dynamically transition learners from Easy to Medium and Hard items.
4) *Full-length timed mock examinations:* Adding countdown timers, question palettes with Mark for Review states, pause/resume persistence, and syllabus-weighted mock blueprints will simulate real examination-hall conditions.
5) *Field evaluation:* Future studies will evaluate learning gains across a cohort of JEE, NEET, and GATE aspirants by comparing unassisted post-test scores between students using QGenAI's mistake-grounded AI Tutor and those using static answer keys alone.

VIII. CONCLUSIONS

We presented QGenAI, a full-stack web platform that unifies competitive examination MCQ practice, server-side answer evaluation, account-scoped learning analytics, and mistake-driven AI tutoring for JEE, NEET, and GATE aspirants. By decoupling question delivery from server-side grading, enforcing PostgreSQL Row-Level Security, and grounding LLM tutoring prompts in a learner's actual recorded errors, the system overcomes the fragmentation of conventional study workflows. The worked evaluation across 28 sessions demonstrates how difficulty breakdowns and early-signal mistake queues guide targeted revision, establishing a solid foundation for future adaptive testing and RAG-based question generation.

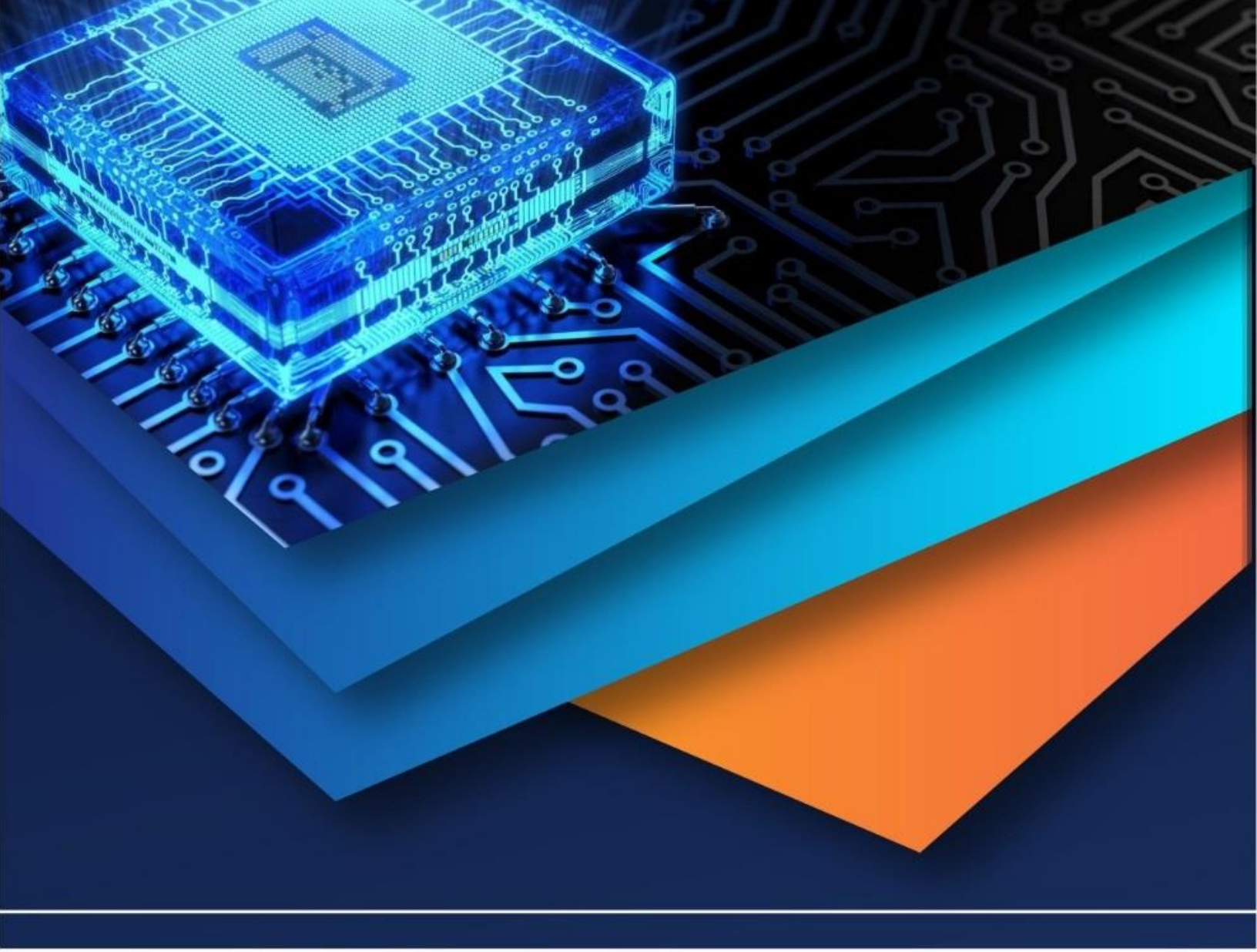
IX. ACKNOWLEDGMENT

The authors express their sincere gratitude to Dr. R. Radhakrishnan, M.E., Ph.D., Principal, and Dr. G. Fathima, M.E., Ph.D., Professor and Head of the Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, for providing the facilities and academic environment for this work. The authors are especially indebted to their supervisor, Mrs. S. Mohana Preetha, M.E., Assistant Professor, Department of CSE, for her continuous guidance, technical reviews, and encouragement throughout this project.

REFERENCES

[1] F. O. Egara and C. Mosia, "Effectiveness of AI-based tutoring and assessment systems in mathematics education: A systematic review," Discover Educ., vol. 5, 2026.
[2] X. Wang, Y. Maeda, and H.-H. Chang, "Development and techniques in learner model in adaptive e-learning system: A systematic review," Comput. Educ., vol. 225, Art. no. 105184, Feb. 2025.
[3] M. Al-Emran and S. I. Malik, "A review on the use of large language models as virtual tutors," Sci. Educ., vol. 34, no. 2, pp. 877–892, Apr. 2025.
[4] H. Bastani, O. Bastani, A. Sungu, H. Ge, Ö. Kabakcı, and R. Mariman, "Generative AI without guardrails can harm learning: Evidence from high school mathematics," Proc. Nat. Acad. Sci. USA, vol. 122, no. 26, Art. no. e2422633122, Jul. 2025.
[5] Z. Shou, M. Xie, J. Mo, and H. Zhang, "Predicting student performance in online learning: A multidimensional time-series data analysis approach," Appl. Sci., vol. 14, no. 6, Art. no. 2522, Mar. 2024.
[6] N. Daheim, J. Macina, M. Kapur, I. Gurevych, and M. Sachan, "Stepwise verification and remediation of student reasoning errors with large language model tutors," in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), Miami, FL, USA, Nov. 2024, pp. 8386–8411.
[7] Z. A. Pardos and S. Bhandari, "ChatGPT-generated help produces learning gains equivalent to human tutor-authored help on mathematics skills," PLOS ONE, vol. 19, no. 5, Art. no. e0304013, May 2024.
[8] E. Kasneci et al., "ChatGPT for good? On opportunities and challenges of large language models for education," Learn. Individ. Differ., vol. 103, Art. no. 102274, Apr. 2023.
[9] M. Riestra-González, M. del P. Paule-Ruíz, and F. Ortin, "Massive LMS log data analysis for the early prediction of course-agnostic student performance," Comput. Educ., vol. 163, Art. no. 104108, Apr. 2021.
[10] S. Wollny, J. Schneider, D. Di Mitri, J. Weidlich, M. Rittberger, and H. Drachsler, "Are we there yet? A systematic literature review on chatbots in education," Front. Artif. Intell., vol. 4, Art. no. 654924, Jul. 2021.
[11] C. Yang, L. Luo, M. A. Vadillo, R. Yu, and D. R. Shanks, "Testing (quizzing) boosts classroom learning: A systematic and meta-analytic review," Psychol. Bull., vol. 147, no. 4, pp. 399–435, Apr. 2021.
[12] D. Ifenthaler and J. Y.-K. Yau, "Utilising learning analytics to support study success in higher education: A systematic review," Educ. Technol. Res. Dev., vol. 68, no. 4, pp. 1961–1990, Aug. 2020.
[13] G. Akçapınar, A. Altun, and P. Aşkar, "Using learning analytics to develop early-warning system for at-risk students," Int. J. Educ. Technol. High. Educ., vol. 16, Art. no. 40, Oct. 2019.
[14] O. Viberg, M. Hatakka, O. Bälter, and A. Mavroudi, "The current landscape of learning analytics in higher education," Comput. Hum. Behav., vol. 89, pp. 98–110, Dec. 2018.
[15] O. O. Adesope, D. A. Trevisan, and N. Sundararajan, "Rethinking the use of tests: A meta-analysis of practice testing," Rev. Educ. Res., vol. 87, no. 3, pp. 659–701, Jun. 2017.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)