International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

AI-Enhanced Question Paper Generator Using Ant Colony Optimization

Prof. Sanjay Kalamdhad[1], Mr. Smit Bhusari[2], Mr. Sannidhya Bhaisare[3], Mandar Patne[4], Vedant Ingale[5]

[1]Assistant Professor, [2,3,4,5]Student
[1,2,3,4,5]Department of Computer Science and Business Systems
St. Vincent Pallotti College of Engineering and Technology
Nagpur, India
[1]kalam.sanjay@gmail.com, [2]smitbhusari.22@stvincentngp.edu.in,
[3]sannidhyabhaisare.22@stvincentngp.edu.in, [4]mandarpatne.23d@stvincentngp.edu.in,
[5]vedantingale.22@stvincentngp.edu.in

Abstract
Creating examination question papers is a timeconsuming and repetitive manual task for educators. This process is prone to inconsistencies in topic coverage, difficulty distribution, and fairness. This paper presents an innovative system, an ”AI-Enhanced Question Paper Generator,” designed to automate and optimize this process. The system introduces a hybrid approach by integrating Generative AI with the Ant Colony Optimization (ACO) algorithm. The Generative AI, powered by the Google Gemini API, dynamically analyzes and extracts content from any provided study material (e.g., text, PDF) to generate a diverse pool of questions. This fundamentally eliminates the reliance on static, manually curated question banks, allowing the system to adapt to new syllabi and materials instantly. Subsequently, the ACO algorithm, a bio-inspired metaheuristic, intelligently selects the optimal set of questions from this pool. The optimization process ensures the final question paper adheres to predefined constraints, including balanced difficulty levels, comprehensive topic coverage, and precise marks distribution. The proposed system drastically reduces the workload for educators, enhances the quality and fairness of assessments, and provides a scalable solution for educational institutions.

Index Terms—Generative AI, Ant Colony Optimization, Automated Assessment, Educational Technology (EdTech), Natural Language Processing (NLP), Question Paper Generation

I. INTRODUCTION

A. The Challenge of Modern Assessment

In the modern era of education, technology has become a crucial enabler of innovation, efficiency, and accessibility [1]. From digital classrooms to adaptive e-learning platforms, the role of technology in transforming teaching and learning practices is undeniable. Among the many academic activities,

assessment plays an integral role in measuring knowledge, evaluating skills, and ensuring learning outcomes are met.

Despite its importance, the process of creating examination question papers has remained largely manual and laborintensive [2]. Teachers and faculty members are required to carefully design papers by selecting questions from textbooks or existing question banks, while ensuring a balanced coverage of topics, levels of difficulty, and distribution of marks. This process is not only time-consuming but also repetitive, leaving educators with less time for actual teaching and student engagement.

Over the past decade, several digital solutions have been introduced for automating paper generation. However, these systems are often restricted to static question banks [3]. Such tools require educators to manually enter and store questions. With frequent changes in curricula and the dynamic nature of knowledge resources, manual updates to these banks become both tedious and impractical. Consequently, the scope and efficiency of current solutions remain constrained.

B. The Role of Generative AI in Education

Generative AI, an advanced subset of AI, has gained tremendous attention for its ability to produce contextually meaningful and creative outputs from raw data [6]. In the educational domain, Generative AI can be used to automatically generate diverse learning resources, summaries, and assessment items [8].

By analyzing study material in the form of text, PDFs, or digital documents, Generative AI is capable of formulating relevant questions aligned with the content. This project specifically leverages the Google Gemini API as its core Generative AI engine. This integration eliminates the dependency on pre-stored question banks and ensures the system remains adaptable to any new material. When a new textbook is introduced or a syllabus is revised, the AI can directly process the updated content and produce fresh sets of questions without manual intervention [4]. This flexibility allows it to create a wide range of question types, from Multiple Choice Questions (MCQs) to descriptive and analytical questions, catering to diverse examination needs [9].

C. Ant Colony Optimization: A Bio-Inspired Approach

While Generative AI can efficiently produce questions, the challenge lies in selecting and structuring them into a balanced exam paper. This is where Ant Colony Optimization (ACO) comes into play. Inspired by the foraging behavior of real ants, ACO is a metaheuristic algorithm widely applied to combinatorial optimization problems such as routing, scheduling, and resource allocation [5].

In this context, ACO treats each candidate question paper as a potential solution. Questions are represented as nodes, and an artificial ”ant” traverses these nodes to build a complete paper. The selection of questions is guided by pheromone trails—mathematical values that encode the desirability of specific paths based on prior evaluations. Over successive iterations, pheromone intensities are updated, reinforcing highquality solutions.

The algorithm evaluates candidate papers based on multiple constraints, such as difficulty distribution (easy, medium, hard), marks allocation, question type diversity (MCQ, short answer), and topic coverage. By simulating the collective intelligence of ant colonies, the system gradually converges toward an optimal paper that satisfies all requirements [7].

D. Technology Integration and Contribution

The innovation of this project lies in the synergy between Generative AI and Ant Colony Optimization. While AI ensures dynamic question creation, ACO provides intelligent optimization for paper structuring. Together, they address both adaptability and fairness—two critical challenges in assessment design.

The implementation is supported by a robust technology stack. The frontend is designed using HTML, CSS, and

JavaScript to provide an intuitive user interface. The backend, built with Java and Python using the Spring Boot framework, manages the system logic, AI integration, and optimization processes. A MySQL database ensures secure storage of user information and generated papers. This integration not only ensures smooth functionality but also makes the platform future-ready and scalable for institutional use [10].

II. PROBLEM STATEMENT

Creating examination question papers is one of the most repetitive and time-consuming tasks for educators. Teachers are required to manually select and arrange questions while ensuring coverage of the syllabus, appropriate difficulty distribution, and fair allocation of marks. This manual process is highly demanding and often results in inconsistencies, unintentional bias, and significant consumption of valuable teaching hours.

Existing automated systems provide only limited relief [1]. Most of these solutions rely heavily on pre-defined, static question banks, which require continuous manual data entry and updates [3]. When new study materials such as textbooks, lecture notes, or academic PDFs are introduced, these systems cannot automatically adapt, making them unsuitable for dynamic academic environments.

Another major challenge lies in the lack of intelligence in maintaining balance and fairness [16]. Current tools do not guarantee uniform distribution of topics and difficulty levels, often leading to over-representation or under-representation of certain areas. Repetition of questions is a recurring issue, while ensuring proper alignment with learning objectives remains a significant concern. Such inconsistencies compromise the reliability of assessments and limit their effectiveness in measuring true student learning outcomes.

A. Objectives

To address the aforementioned problems, this project aims to achieve the following objectives:

- Automated Question Generation: Develop a system that uses Generative AI (Google Gemini API) to read and understand any study material (text, PDF, or notes) and generate relevant questions in multiple formats (MCQs, short-answer, descriptive).
- Optimized Question Selection: Implement the Ant Colony Optimization (ACO) algorithm to intelligently select questions based on difficulty levels, topic coverage, marks distribution, and fairness, ensuring a balanced and high-quality exam paper.
- User-Friendly Web Interface: Create an intuitive, accessible web application that allows teachers to easily upload study materials, configure paper requirements, and instantly generate printable question papers without technical expertise.
- Efficiency and Fairness: Reduce the manual workload for educators, eliminate bias in question selection, and ensure exam papers consistently meet academic standards and syllabus requirements.

III. RELATED WORK

Automated Question Generation (AQG) has evolved from rule-based systems to deep learning-driven models. Rule-based methods generate grammatically correct but often repetitive questions, while neural models like T5-QG and BERT-QG introduce greater semantic diversity [17]. A comprehensive survey by Islam et al. [4] analyzed various NLP-based methods, highlighting the need for intelligent systems capable of processing new content without pre-tagged datasets.

Several researchers have proposed systems for automatic question paper generation. Shendure et al. [1] and Nalawade [3] implemented systems that use a semantically tagged question repository. While effective in reducing teacher workload, these systems are fundamentally limited by their reliance on a static question bank, making them unable to adapt to new study materials in real-time. Gangar et al. [2] focused on improving paper balance regarding syllabus coverage and difficulty, but their approach still required manual question entry.

In parallel, optimization techniques have been explored for paper structuring. Genetic Algorithms and Particle Swarm Optimization have been used; however, Ant Colony Optimization (ACO) often offers a better exploration-exploitation balance for combinatorial problems [15]. The seminal work by Dorigo and Gambardella [5] on ACO provides the theoretical foundation for our optimization layer.

Our project builds upon these existing works by uniquely integrating state-of-the-art Generative AI (Google Gemini API) with Ant Colony Optimization. Unlike systems dependent on pre-built repositories, our approach dynamically generates questions from any uploaded material and then optimizes their selection to produce balanced, fair, and highquality exam papers [13].

IV. PROPOSED SYSTEM ARCHITECTURE

The system is designed with a modular, layered architecture to ensure scalability and maintainability. It separates the concerns of user interaction, AI-based content generation, optimization, and data storage. The architecture is best understood through its workflow and structural components.

A. System Workflow Model

The end-to-end process of generating a question paper follows a clear, four-step workflow, as illustrated in Fig. ??.

1) Step 1: Send Request: The Teacher/Educator accesses the web application interface. They upload the study material (e.g., PDF or text file) and specify the constraints for the paper (e.g., total marks, number of questions, difficulty ratio). This request is sent to the system backend.
2) Step 2: Create Questions: The backend service forwards the study material to the AI Generation Service, which is powered by the Google Gemini API. The AI service analyzes the content and generates a large, diverse pool of questions (MCQs, short answer, etc.) relevant to the material.
3) Step 3: Select Best Questions: This large pool of questions is fed into the ACO Selection Service. The Ant Colony Optimization algorithm runs, treating each potential paper as a ”path” and evaluating it against the user-defined constraints. It iteratively converges on an ”Optimized Set of Questions.”
4) Step 4: Format the Paper: The optimized set of questions is passed to the paper formatting module. This module arranges the questions into a structured, professional document, which is then delivered to the user as the Final Question Paper (in PDF or DOCX format).

B. System Architecture

The system's components are organized into distinct layers, as shown in Fig. ??.

- User Layer: This is the frontend, consisting of the
  User Interface (Web App) built with HTML, CSS, and JavaScript (ReactJS). It is the educator's sole point of interaction, used for uploading materials and configuring constraints.
- AI Layer: This layer contains the core intelligence for content creation. When material is uploaded, it is sent to the AI Question Generator (Google Gemini API), which processes the text and returns a large set of generated questions.
- Optimization Layer: The generated questions are passed to this layer. The ACO Optimization Module evaluates each question based on its metadata (difficulty, topic) and the user's constraints, selecting the optimal set.
- Storage and Formatting: The optimized question set is stored in the Database (MySQL) for record-keeping and potential reuse. It is simultaneously sent to the Output Formatting Module, which assembles the final formatted document for the user.

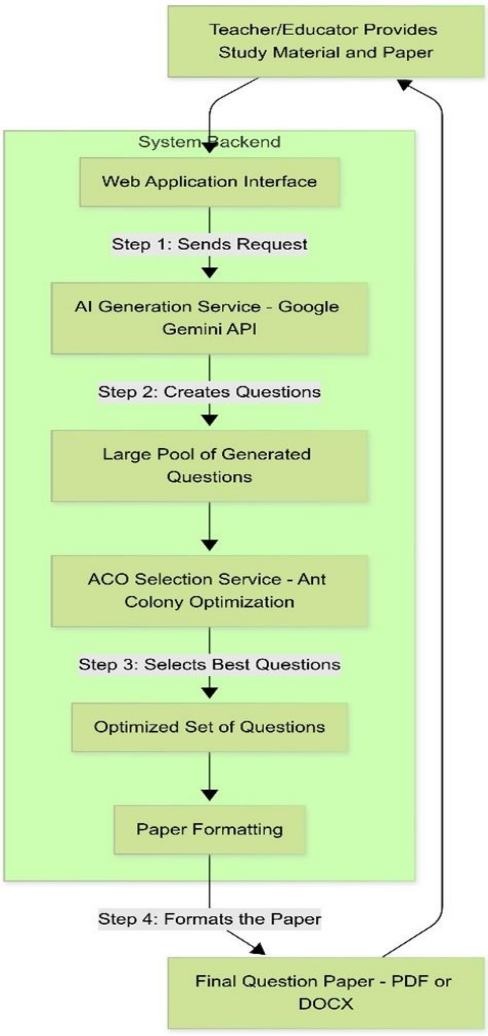

Fig. 1. Workflow Diagram of the AI-Enhanced Question Paper Generator, detailing the 4-step process from request to final output.

V. METHODOLOGY AND IMPLEMENTATION

The successful execution of the system relies on the intricate collaboration between the AI generation module and the ACO selection module.

A. AI-Driven Question Generation

This module employs prompt-based interaction with the Google Gemini API. When the educator uploads study material, the backend system first parses the content (using tools like Apache Tika for PDF and DOCX) into clean text. This

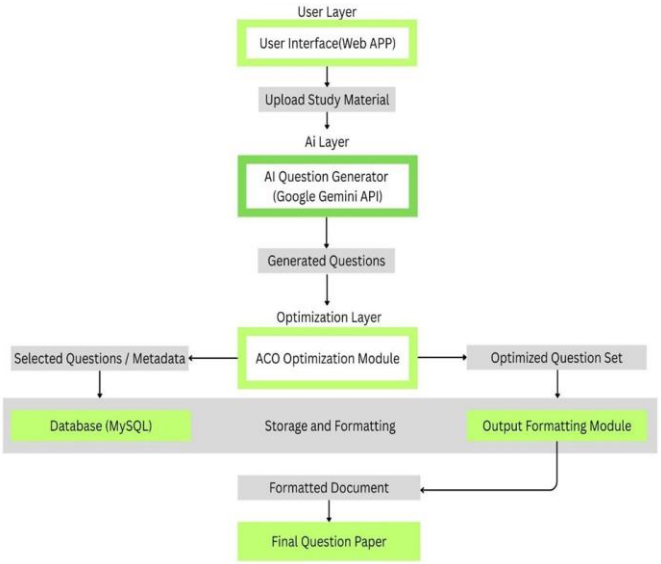

Fig. 2. System Architecture

text is segmented and used to engineer specific prompts for the AI model. The prompts instruct the AI to generate questions of various types based on the provided text, such as:

- Multiple Choice Questions (MCQs): Conceptual questions with plausible distractors.
- Short Answer Questions: Factual or definitional recall.
- Descriptive Questions: Analytical or problem-solving prompts.

Each question generated by the AI is also tagged with predicted metadata, including difficulty (easy, medium, hard), estimated marks, and associated topic keywords.

B. ACO-Based Question Selection

This is the core optimization phase. Let $Q = \{q_1, q_2, \ldots, q_n\}$ represent the entire pool of questions generated by the AI. A valid question paper is a subset $P \subseteq Q$ that must satisfy a set of constraints C (e.g., total marks M, difficulty ratio D, topic coverage T).

The quality of any candidate paper P is evaluated by a fitness function F(P). This function is designed to reward adherence to constraints and penalize deviations. A generalized form of the fitness function is:

$$F(P) = w1Ccoverage + w2Cdifficulty - w3Ppenalty \quad (1)$$

where w_i are weights, $C_{coverage}$ measures topic distribution, $C_{difficulty}$ measures the accuracy of the easy-medium-hard ratio, and $P_{penalty}$ enforces strict limits on total marks and question counts.

Artificial ”ants” iteratively build solutions (papers) by selecting questions. The probability p_i of an ant selecting question q_i is based on the pheromone level τ_i (historical success of this question) and the heuristic desirability η_i (how well this question fits the current paper being built):

$$p_i = \frac{(\tau_i)^\alpha (\eta_i)^\beta}{\sum_j (\tau_j)^\alpha (\eta_j)^\beta} \quad (2)$$

where α and β are parameters that control the influence of pheromones versus heuristic information.

After each iteration (where all ants have built a paper), the pheromone trails are updated. The trails on questions belonging to the best-performing papers are reinforced, while all trails undergo slight ”evaporation” (ρ):

$$\tau_i = (1-\rho)\tau_i + \Delta\tau_i \quad (3)$$

where $\Delta\tau_i$ is the amount of pheromone deposited, proportional to the fitness F(P) of the best paper containing q_i. This process allows the algorithm to converge on a high-quality, balanced question paper.

C. Algorithmic Flow

The combined process is summarized in the pseudocode in Algorithm 1.

Algorithm 1 AI-ACO Question Paper Generation

Input: StudyMaterial S, Constraints C
Output: FinalQuestionPaper P_{best}
// AI Generation Phase
TextData ← Parse(S) QuestionPool Q ←∅ for segment in TextData do
Q ← Q∪ GeminiAPI.GenerateQuestions(segment) end for // ACO Selection Phase
InitializePheromones(Q)
P_{best} ←∅
F_{best} ← 0 for iteration = 1 to MaxIterations do
for ant = 1 to NumAnts do
$P_{current}$ ←∅
while not ConstraintsSatisfied($P_{current}$,C) do
q_{next} ← SelectQuestion(Q,τ,η)
Pcurrent ← Pcurrent ∪{qnext} end while
Fcurrent ← CalculateFitness(Pcurrent) if Fcurrent > Fbest then
Pbest ← Pcurrent
Fbest ← Fcurrent end if
end for
UpdatePheromones(τ,P_{best}) end for
// Formatting Phase FormattedPaper ← Format(P_{best}) return FormattedPaper

TABLE I
PERFORMANCE COMPARISON OF QUESTION PAPER GENERATION METHODS

Method	Coverage Score	Difficulty Match	Time (min)
Manual Creation	0.75	0.68	120
Random Selection	0.70	0.55	10
Proposed AI+ACO	0.92	0.83	5

D. Implementation Details

The system is developed using a modern technology stack to ensure robustness and separation of concerns.

- Frontend: ReactJS (using HTML, CSS, JavaScript) for a responsive, component-based user interface [19].
- Backend: Spring Boot (Java) to handle business logic, API routing, and user management. A Python-based microservice manages the AI and ACO components.
- Database: MySQL for storing user profiles, system configurations, and generated question paper metadata.
- Content Extraction: Apache Tika for parsing various document formats (PDF, DOCX) into plain text [18].
- Core Intelligence: Google Gemini API for the Generative AI module.
- Optimization: A custom Ant Colony Optimization algorithm implemented in Python.
- Development Tools: Visual Studio Code, Git for version control, and Postman for API testing.

VI. RESULTS AND EVALUATION

A. Experimental Setup

The system was tested using study materials from Computer Engineering subjects, including ”Data Structures” and ”Internet of Things (IoT).” The uploaded course notes for each subject were processed by the AI module, which generated a pool of 250-300 candidate questions per subject.

To evaluate performance, we used the following metrics:

- Coverage Score: The proportion of unique syllabus topics represented in the final paper.
- Difficulty Match: The accuracy of the generated paper in matching a targeted easy-medium-hard ratio (target: 40%-40%-20%).
- Time Efficiency: The total time in minutes required to generate a complete, formatted paper from the moment of request.

We compared the performance of our ”Proposed AI+ACO” system against two baselines: ”Manual Creation” (based on faculty interviews) and ”Random Selection” (a script that randomly pulls questions from the AI-generated pool).

B. Quantitative Analysis

The comparative performance is presented in Table I. The proposed hybrid AI+ACO approach demonstrates superior performance across all metrics.

The results indicate that the AI+ACO system achieved a 0.92 coverage score, ensuring that 92% of the specified topics were included, a significant improvement over the 75% from manual curation. Furthermore, its ability to match the difficulty ratio (0.83) was 22% higher than the manual method. Most notably, the time efficiency improved by a factor of 24, reducing a 2-hour (120 min) task to just 5 minutes.

C. Qualitative Evaluation

Faculty reviewers were asked to rate the papers generated by the AI+ACO system. The feedback was overwhelmingly positive. Reviewers rated the papers as ”well-structured,” ”varied,” and ”highly syllabus-aligned.” They particularly appreciated the system’s dynamic adaptability, noting its ability to generate valid papers for new syllabi immediately. Some reviewers suggested future enhancements, including validation filters for question clarity and uniqueness detection [14].

VII. DISCUSSION

The hybrid AI-ACO model successfully merges creative, context-aware content generation with structured, constraintbased optimization. The key benefits of this model are:

- Adaptability: The system is not bound by a question bank and can work for any subject or new syllabus provided in text format.
- Scalability: The architecture is easily deployable for large institutions, handling multiple requests concurrently.
- Objectivity: The ACO module’s quantitative balancing ensures fairness and consistency, reducing the ”humanin-the-loop” bias.

Limitations of the current system involve a reliance on the online Google Gemini API, which requires a stable internet connection, and the potential for semantic redundancy in the AI-generated content.

A. Superiority over General AI Tools

A valid question is why this complex system is necessary when a general AI tool like Gemini could be prompted directly. The ”Need for the Proposed System” becomes clear when comparing workflows and outcomes.

- Controlled & Balanced Question Selection: A pure AI, if prompted to ”create a 50-mark paper,” will generate questions, but it will not reliably guarantee balanced topic coverage, marks distribution, or the precise easy/medium/hard ratio. Our system automatically enforces these constraints using the ACO algorithm, a task that pure AI alone does not handle reliably.
- Quality Consistency & Fairness: Direct AI output can be inconsistent—sometimes too easy, sometimes too hard. Our system applies a rigorous optimization and filtering layer to ensure consistency and fairness across all generated papers.
- Time-Saving vs. Prompt-Engineering: Using a general AI tool requires an educator to craft detailed, iterative prompts and then manually check the output. Our system ”needs no prompt crafting.” The user uploads material, sets simple constraints via a GUI, and receives a final, formatted paper instantly.

- Institutional Requirements Compliance: Schools and colleges often have strict formatting and structural patterns for exams. Our system is designed to ”hard-code” compliance with such formats. A general AI tool will not know or follow an institute’s specific pattern.
- Integrated Workflow: Our system integrates question generation, intelligent selection, formatting, and ready-toprint output into one seamless workflow. General AI tools only provide raw text, requiring the educator to manually filter, arrange, and format the paper afterward.

VIII. ETHICAL CONSIDERATIONS AND EDUCATIONAL IMPACT

While automation aids efficiency, it must be used responsibly. Generative AI tools may inadvertently reproduce biased or inaccurate content from their training data. Therefore, a human-in-the-loop approach is essential. Our system designates the educator as the final reviewer; instructor review remains an integral step before publication and administration of the exam.

The educational impact is significant. The system enhances fairness by standardizing difficulty and ensuring equal topic coverage across all exam sessions [16]. Furthermore, it reduces the administrative workload for teachers, enabling them to dedicate more time to instructional design, mentoring, and personalized student feedback. Institutions can maintain consistent evaluation quality, quickly regenerate papers for remedial exams, and adapt assessments for varying cognitive levels using the configurable parameters.

IX. CONCLUSION AND FUTURE WORK

This research introduced an automated question paper generator that successfully integrates Generative AI (Google Gemini API) with Ant Colony Optimization. The system minimizes human intervention while maintaining syllabus compliance, difficulty balance, and question variety. The results demonstrate that this hybrid approach is not only 24 times faster than manual methods but also produces papers with superior topic coverage and difficulty balancing. By eliminating the static question bank, it offers a truly adaptive and scalable solution for modern educational institutions.

The ”Scope of the Project” section in our initial synopsis explicitly noted that the current version would not cover certain features, which now form the basis for our future work. Future directions include:

- AI-Based Answer Key Generation: Extending the AI module to automatically generate answer keys and detailed evaluation rubrics for the selected descriptive questions [12].
- LMS Integration: Developing plugins for direct integration with popular Learning Management Systems (LMS) like Moodle or Canvas, allowing for seamless workflow from course content to assessment.
- Multilingual Question Generation: Leveraging the multilingual capabilities of modern AI models to generate question papers in various languages, enhancing accessibility.
- Duplicate Detection: Implementing automated semantic duplicate detection to ensure question variety across successive exams [11].

This approach demonstrates how hybrid AI models can transform traditional academic workflows into intelligent, efficient, and fair data-driven systems.

ACKNOWLEDGMENT

The authors thank their project guide, Prof. Sanjay Kalamdhad, and the faculty at St. Vincent Pallotti College of Engineering and Technology, Nagpur, for their invaluable guidance and mentorship throughout this project.

REFERENCES

1. A. S. Shendure, A. D. Khamkar, P. P. Vathare, and A. Salavi, ”Question Paper Generator System,” International Journal of Advanced Research in Computer and Communication Engineering, vol. 11, no. 5, pp. 12-17, May 2022.
2. F. K. Gangar, H. G. Gori, and A. Dalvi, ”Automatic Question Paper Generator System,” ResearchGate, 2021.
3. G. Nalawade, ”Automatic Generation of Question Paper from User Entered Specifications using a Semantically Tagged Question Repository,” International Journal for Scientific Research & Development, vol. 8, no. 3, pp. 45-49, 2020.
4. S. M. F. Islam, M. F. Mridha, et al., ”A Survey on Automatic Question Generation from Text,” IEEE Access, vol. 9, pp. 22986-23006, Feb. 2021, doi: 10.1109/ACCESS.2021.3055941.
5. M. Dorigo and L. M. Gambardella, ”Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem,” IEEE Transactions on Evolutionary Computation, vol. 1, no. 1, pp. 53-66, Apr. 1997, doi: 10.1109/4235.585892.
6. A. K. Bhowmick, R. Ganguly, and S. Mondal, ”Automating Question Generation from Educational Text,” arXiv preprint arXiv:2309.15004, Sep. 2023.
7. Y. Shang, Z. Wu, H. Sun, and L. Zhang, ”Reinforcement Learning Guided Multi-Objective Exam Paper Generation,” arXiv preprint arXiv:2303.01042, Mar. 2023.
8. S. Bulathwela, H. Muse, and E. Yilmaz, ”Scalable Educational Question Generation with Pre-trained Language Models,” arXiv preprint arXiv:2305.07871, May 2023.
9. K. Vachev, V. Nikolov, and I. Koychev, ”Leaf: Multiple-Choice Question Generation,” arXiv preprint arXiv:2201.09012, Jan. 2022.
10. N. Vyas, H. Kothari, A. Jain, and A. R. Joshi, ”Automated Question and Test-Paper Generation System,” International Journal of Computer Aided Engineering and Technology, vol. 14, no. 3, pp. 285-300, Mar. 2022, doi: 10.1504/IJCAET.2022.122152.
11. P. G. Naik and K. S. Oza, ”Version Control System for Auto Generation of Question Papers,” in Smart Computing and Communication Systems (SMART 2023), Springer, pp. 355-366, Jun. 2023, doi: 10.1007/978-98199-0769-4 33.
12. N. A. Vishwakarma, S. Patel, and R. Sharma, ”EduQGen: Al-Based Question Paper Generation System,” International Journal of Scientific Research in Engineering and Management (IJSREM), vol. 9, no. 6, pp. 112-118, Jun. 2025.
13. K. P. Kumar, S. B. Reddy, and M. M. Rao, ”Automated Question Paper Generator Using LLM,” International Journal of Research and Innovation in Applied Science (IJRIAS), vol. 10, no. 5, pp. 85-89, May 2025.
14. H. P. S., V. K. R., and S. S., ”System for Automatic Generation of Question Papers Simplifying Assessment and Evaluation,” International Journal of Advanced Research in Computer Networking, Wireless and Mobile Communications, vol. 13, no. 4, pp. 25-30, Apr. 2025.

15. S. B. Bhonde, S. Jejurkar, and S. Galande, ”Automatic Question Paper Generation Using Machine Learning Approach,” International Journal of Innovations in Engineering Research and Technology (IJIERT), vol.
16. 8, no. 3, pp. 1-6, Mar. 2021.
17. S. K. Singh, A. Kumar, and P. Srivastava,”Towards Development of a System for Automatic Assessment of the Quality of a Question Paper,” Smart Learning Environments, vol. 8, no. 6, pp. 1-20, Mar. 2021, doi: 10.1186/s40561-021-00148-9.
18. S. S. Mucciaccia, F. A. Gonzalez, and R. Silva,”Automatic MultipleChoice Question Generation and Evaluation Systems Based on LLM:
19. A Study Case With University Resolutions,” in Proceedings of the 31st International Conference on Computational Linguistics (COLING 2025), pp. 1825-1837, Jan. 2025.
20. ”Apache Tika,” Apache Software Foundation. [Online]. Available: https: //tika.apache.org/ (Accessed: Oct. 2025).
21. ”React - A JavaScript library for building user interfaces,” Meta.
22. [Online]. Available: https://reactjs.org/ (Accessed: Oct. 2025).