Global Journal of Engineering Innovations & Interdisciplinary Research

Original Article

Multiple Choice Quizzes using Langchain and Gemini LLM

K Sudhakar[1], Sannidhi Venkata Sai Revanth[2], Nayab Sama Sulthana[2], V Purushotham Reddy[2], S Sindhuja[2], K Vannur Swamy[2]

[1]Assistant Professsor, Department of Computer Science and Engineering Gates Institute of Technology , Andhra Pradesh, India.

[2]Department of Computer Science and Engineering Gates Institute of Technology , Andhra Pradesh, India.

Correspondence

Dama Rajeswari
Department of Computer Science and Engineering, Gates Institute of Technology, Andhra Pradesh, India.

- Received Date: 11 Feb 2026
- Accepted Date: 02 Apr 2026
- Publication Date: 25 Apr 2026

Keywords

mcq generator, large language models, natural language processing, google gemini pro, langchain, generative ai.

Abstract

The research study investigates the use of cutting- edge technologies like Langchain and Gemini AI for the automated creation and assessment of multiple-

choice questions (MCQs). Although producing multiple- choice questions (MCQs) has traditionally been done by hand and required a lot of work, advances in artificial intelligence (AI) and natural language processing (NLP) have opened up new possibilities. The creation and implementation of an MCQ generator—which uses Gemini AI to generate MCQs and Langchain for rapid engineering are covered in this paper. Customizing prompts for prompt engineering, chaining prompts with the Gemini AI model, and utilizing OpenAI's GPT-3.5 and multiple Language Learning Models (LLMs) to assess the created MCQs are the steps in the process. The study intends to assess these models' efficiency in handling intricate queries,producing appropriate answers, and examining the caliber of the MCQs that are produced.

Introduction

IIn the dynamic field of education, the creation of Multiple- Choice Questions (MCQs) has traditionally been a manual and labor-intensive task. However, the emergence of Natural Language Processing (NLP) and advancements in Artificial Intelligence (AI) are ushering in a new era.

This research paper delves into the development and application of an MCQ generator, harnessing the power of innovative technologies such as Langchain and Gemini AI.

The purpose of this research is to evaluate the effectiveness of these models in processing complex queries and generating suitable solutions, thereby providing a brief overview of MCQ generation.

A significant aspect of this project is the use of Language Models (LLMs), with a particular focus on Gemini AI, a relatively new model that has not been extensively tested for this specific application. The project aims to analyze the efficiency of this LLM in the specific task of MCQ generation and its ability to handle complex queries and provide appropriate solutions. The research not only generates MCQs but also evaluates these questions, offering a comprehensive understanding of the newer technologies used.

MCQs play a crucial role in education and assessment, offering a nuanced view of the MCQ generation process. This research serves as a testament to the potential of NLP and AI in revolutionizing educational assessments, making them more efficient, customizable, and adaptable to a wide range of subjects and learning objectives. We believe that our findings will aid in informed decision-making for educators, instructional designers, and ed-tech developers who are considering the integration of LLM-powered tools into their teaching and learning environments.

Literature Review

SRecent literature explores AI-driven MCQ generation using transformers like BERT, T5, and GPT, achieving comparable quality to human-authored questions, especially in medical education. Hybrid frameworks blending rule-based and ML approaches are proposed, employing techniques such as ontologies for stem creation and lexical databases for distractors. Neural models with attention to mechanisms are investigated for reading comprehension-based questions. Techniques like extractive summarization, keyword extraction, and similarity scoring automate MCQ generation, which is evaluated via expert reviews and language model metrics. Utilizing large language models and NLP, these advancements enhance educational assessment automation, yet challenges persist, prompting further research.

The most relevant papers for this research on automated generation and evaluation of multiple-choice quizzes are:

Citation: Sudhakar K, Sannidhi VSR, Nayab SS, Reddy VP, Sindhuja S, Swamy KV. Multiple Choice Quizzes using Langchain and Gemini LLM. GJEIIR. 2026;6(4):249.

Relevant research on automated generation and evaluation of multiple-choice quizzes explores the use of transformer models like BERT to generate questions from input sequences, context paragraphs, and response phrases through finetuning. These models, including variations that integrate sequential data and context/answer tokens, have been shown to improve question quality and reduce ambiguity in the generated content [1].

Further studies compare the quality of MCQs generated by AI models, such as ChatGPT, to those created by human experts, particularly in specialized fields like medical education. While human-generated questions often score higher on relevance, the overall quality of AI-generated MCQs demonstrates the potential of these models to perform on par with human efforts, showing no significant differences in key areas such as difficulty and clarity [2].

Additional work focuses on using transformer models, such as T5, in an end-to-end MCQ generation pipeline. This approach covers the entire process, from parsing and generating question-answer pairs to distractor creation, similarity scoring, and final question preparation, showcasing the capabilities of such models in automating comprehensive quiz generation [3]. Several other papers surveyed different techniques and approaches for automated MCQ generation from text data. Some utilized natural language processing methods like summarization, keyword extraction, distractor generation [4- 7], and concept hierarchy extraction [8]. Other studies focused on neural question generation models with attention mechanisms [9] or leveraged large language models like GPT- 4 [10] and BERT [11] for specialized domains like programming education. Some studies proposed hybrid approaches combining rule-based and learning-based methods as shown in [11]. Evaluating the quality of generated MCQs through expert reviews, language model scoring, and metrics was also explored [12].

A few surveys reviewed general frameworks and systems for automatic MCQ generation [13-15], covering techniques used in different components like question stem generation using ontologies or machine learning. The application of transformer language models for question generation through prompt engineering was also investigated [17-19]. Human-AI comparative studies [20] highlighted the potential of generative AI models to produce high-quality MCQs comparable to human-authored ones while identifying persisting challenges like generating coherent distractors.Overall, the literature demonstrates the growing research interest and progress in automating MCQ generation to enhance educational assessments.

Methodology

Our proposed methodology consists of developing an equity research chatbot wherein we give PDF files (consisting of company financials and equity research reports) as input to ourchatbot which then further generates required responses based on provided prompts to it. Following are the steps that weperform from start to end in our chatbot development.

User Interface of MCQ Generator

The MCQ Generator program is a user-friendly tool written using Streamlit in Python as shown in Fig.1. It allows users to submit a PDF, word, or text file, with a maximum size limit of 200MB, from which the MCQs are generated. Users can specify the amount of MCQs they desire, ranging from 2 to 25, and input the subject of the MCQs. The complexity level of the questions can be set from a dropdown menu with options such as “Simple”, “Intermediate”, and “Advanced”. Once all

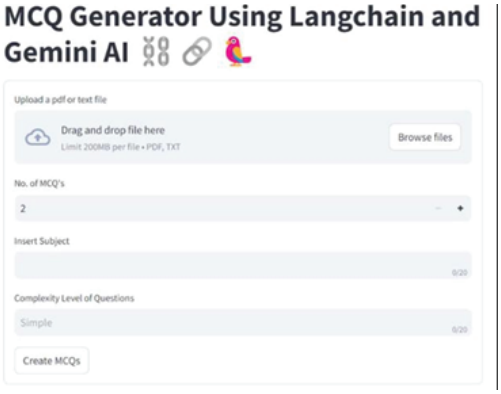

Fig. 1. MCQ Generator User Interface

the essential information is entered, users can click the “Create MCQs” option to generate the MCQs. During the creation process, a loading spinner is presented, giving a responsive user experience. If an error occurs, it is instantly displayed to the user. Upon successful generation, the MCQs are displayed in a table format and a review of the generated MCQs is also supplied in a text section. A unique aspect of this application is the bottom vicinity which includes a complexity analysis and a question similarity section. This gives users a full understanding of the created MCQs, making this program a powerful tool for educators and rest alike.

PDF DataExtraction

It is the first and very most important step of the process where we give PDF data as input to our developed system. After inputting these PDF files, we extract important and required data using PyPDF (a free open-source Python PDF library capable of merging, splitting, cropping, and transforming the pages of PDF files) here more specifically we are first splitting text and then extracting this text as data for further analysis and processing on it.

Prompts

Instead of using the time-consuming and complicated old method of generating multiple-choice questions using natural language processing techniques like summarizing, keyword extraction, and distractor generation. The process of generating MCQs can be made quick, easy, and lesscomplicated by introducing prompt engineering methods.Recent advancements in AI with the help of clear, concise, and specific prompting will help create MCQs more easily than past traditional methods.

To customize prompts more easily we have made use of prompt templates from langchain. Prompt templates in LangChain provide predefined structures and formats to guide users in constructing prompts for language model-based tasks, ensuring clarity and consistency in interactions. These templates include placeholders for input variables, formatting instructions, and guidelines for incorporating context,facilitating the effective communication of task requirements to the language model. In Fig.2 is the generate the multiple- choice questions from Gemini:

The template is structured so that you can customize the text, number of questions desired, subject, difficulty level, and the required format of output in the prompt. The prompt is then chained with the Gemini large

language model and the output key is named “quiz”; this key will contain the generated questions according to the specified JSON format. So, the prompt is sent to the Gemini server using

```
TEMPLATE="""
Text:{text}
You are an expert MCQ maker. Given the above text, it
is your job to \
create a quiz  of {number} multiple choice questions
for {subject} students in {tone} tone.
Make sure the questions are not repeated and check all
the questions to be conforming the text as well.
Make sure to format your response like  RESPONSE_JSON
below  and use it as a guide. \
Ensure to make {number} MCQs
RESPONSE_JSON:
{response_json}
"""
```

Fig. 2. MCQ Generator Prompt

```
{
    "1": {
        "mcq": "multiple choice question",
        "options": {
            "a": "choice here",
            "b": "choice here",
            "c": "choice here",
            "d": "choice here"
        },
        "correct": "correct answer"
    },
    "2": {
        "mcq": "multiple choice question",
        "options": {
            "a": "choice here",
            "b": "choice here",
            "c": "choice here",
            "d": "choice here"
        },
        "correct": "correct answer"
    }
}
```

Fig. 3. JSON Format

the langchain_google_genai API and a response is generated.

The generated responses will be in the JSON format (RESPONSE_JSON) depicted in Fig.3.After the questions are generated, they are sent to a quiz evaluation prompt (Fig.4) that evaluates the complexity of the question and gives a thorough analysis. In this prompt, the LLM is also asked to update the questions that repeated or not at par with the cognitive and analytical abilities of the students.

```
TEMPLATE2="""
You are an expert english grammarian and writer. Given
a Multiple Choice Quiz for {subject} students.\
You need to evaluate the complexity of the question
and give a complete analysis of the quiz. Only use at
max 50 words for complexity analysis.
if the quiz is not at per with the cognitive and
analytical abilities of the students,\
update the quiz questions which needs to be changed
and change the tone such that it perfectly fits the
student abilities
Quiz_MCQs:
{quiz}

Check from an expert English Writer of the above quiz:
"""
```

Fig. 4. Review Prompt

Flow of the Application

The MCQ Generator is a cutting-edge web-based application that leverages state-of-the-art natural language processing (NLP) techniques and advanced language models to facilitate the automated creation and evaluation of high- quality, subject-specific multiple-choice questions (MCQs) based on a given text. This innovative solution represents a significant advancement in the field of educational technology, incorporating several cutting-edge technologies and frameworks.

At the core of the MCQ Generator lies the integration of the Gemini-pro language model from Google's Generative AI platform. This advanced transformer-based model enables the application to generate context-aware MCQs that closely align with the source material and the specified subject area, accurately capturing the nuances and complexities of the given text. By harnessing the power of this state-of-the-art language model, the application ensures a higher level of educational value and relevance in the generated questions.

The application leverages the Streamlit framework, a modern and powerful tool for building interactive data applications in Python. Streamlit's intuitive and user-friendly interface allows for seamless integration of various components, including file uploads, user inputs, and dynamic displays of generated content. This framework simplifies the development process and enhances the overall user experience, making the MCQ Generator accessible and engaging for educators and learners alike.

The workflow of the MCQ Generator is as follows: User Input: The application presents an intuitive interface where users can upload source material (PDF or text files), specify the desired number of MCQs, indicate the subject area, and select the complexity level or tone of the questions.

Text Extraction: Utilizing robust file handling techniques, the application extracts the text content from the uploaded source material, supporting both PDF and text file formats.

MCQ Generation: Leveraging the power of the Gemini-pro language model, the application generates a set of multiple-choice questions based on the extracted text, the specified number of MCQs, the subject area, and the desired complexity level. Complexity

Evaluation: To ensure the generated MCQs align with the cognitive and analytical abilities of the target audience, the application employs another instance of the Gemini-pro language model to evaluate the complexity of the questions. This evaluation takes the intended audience, suggesting modifications or adjustments to the questions or their complexity level as needed.

Review Generation: In addition to the MCQs, the application generates a comprehensive review, providing an in-depth analysis of the overall quality and suitability of the questions. This review leverages the language model's capabilities to provide insightful feedback and highlight potential areas for improvement.

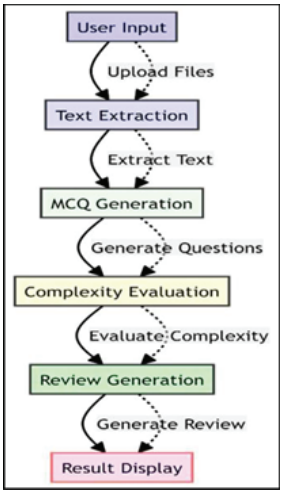

Fig. 5. Flow of Application

Result Display: The generated MCQs and the corresponding review are presented to the user through an interactive and visually appealing interface facilitated by Streamlit. The MCQs are displayed in a tabular format, with each question accompanied by its options and the correct answer, while the review is presented in a dedicated text area, allowing for easy assessment and evaluation.

System Architecture

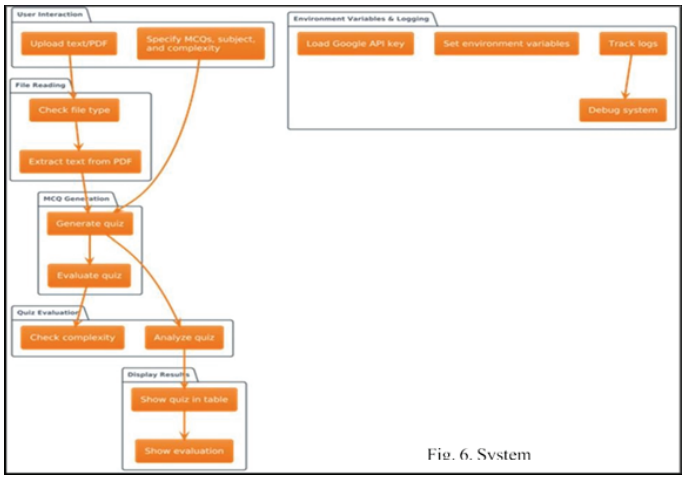

Fig. 6. System

Fig.6 illustrates a system for automated quiz creation and assessment, leveraging user inputs and file uploads. The

components encompass installation, AI-powered multiple-choice question (MCQ) generation, evaluation, file reading, environment/logging, and output for showing results. This system architecture facilitates quiz generation and evaluation:package Installation: Sets up the system, promptinguser inputs like question quantity, subject, and complexity. Also handles file uploads. MCQ Generation: Uses UsePromptTemplate and UseChainGoogleGenerativeAI to create MCQs, forming quizzes via a quiz_chain and assessing them with a review_ chain. Quiz Evaluation: Assesses quiz complexity and offers analysis. File Reading: Extracts text from uploaded PDF or text files. Environment Variables: Manages Google API key and user security. Logging: Tracks activities and logs them to a file Additional components display quizzes and evaluation results. Users can generate and assess quizzes based on uploaded content and specified configurations.

Implementation Details (Programming Languagesand Tools Used)

The MCQ generator is a sophisticated software solution implemented in Python, which leverages a range of frameworks and technologies to achieve its objectives. The standard Python logging module is used for generating log files to track events throughout the software's runtime, aiding in debugging and understanding the software's execution flow. The os and datetime modules are used for interfacing with the operating system and manipulating dates and times, respectively, which are required for managing files and directories, and for timestamping events. The json module is used for interacting with JSON data, a common data interchange format. The pandas library, a powerful data manipulation and analysis tool, is used to extract data and store it in a desired format. The traceback module is used for extracting, formatting, and publishing stack traces of Python programs, which is vital for determining the source of failures.

The software also uses the dotenv module to read key- value pairs from a .env file and add them to the environment variable, a standard approach for managing configuration settings securely and flexibly. The PyPDF2 library is used for reading PDF files, a common format for text documents. The streamlit library is used to construct a web-based user interface for the MCQ generator, giving an interactive and user-friendly manner for users to input their requirements and examine the created MCQs

The heart of the MCQ generation process is the ChatGoogleGenerativeAI model (Gemini-pro) from the langchain_google_genai package. This is a language model developed by Google, which employs machine learning techniques to generate human-like text depending on the provided input. In this situation, it is utilized to produce the MCQs based on the input text. The SequentialChain class is used to chain the MCQ generating and evaluation processes together, ensuring a seamless and fast workflow

In addition to producing MCQs, the software also evaluates the generated quiz for the accuracy of the questions by chaining the response quiz into a prompt to OpenAI's GPT- 3.5 and various distinct LLMs. It evaluates how similar the generated questions are to each other and calculates the similarity score of the options to judge the toughness of the questions. These extra elements ensure the quality of the generated MCQs and provide vital insights for further refining.

Finally, the setuptools library is used for packaging the project, specifying the project name, version, author details, and the dependencies required by the project. This makes it easier to distribute and install the software. Abbreviations and Acronyms

Discussion

Objective

Google's the langchain_google_genai library to create human- like text, which is subsequently translated into MCQs. Additionally, the system employs OpenAI's GPT-3.5 to evaluate the accuracy and quality of the generated MCQs. It examines the complexity of the questions, looks for resemblance among them, and calculates the similarity score of the options to gauge question toughness. The results indicate that the system effectively creates high-quality MCQs and evaluates them accurately, fitting well with the research objectives and highlighting the potential of language models in educational technology.

Limitations

While the results are promising, there are key constraints to this study. The quality of the generated MCQs greatly relies on the input text's quality and complexity. The system may need better structured or extremely complicated documents. Additionally, the technology presently supports only English language texts. The similarity and toughness assessments are based on computational methods, which may only sometimes accord with human opinion. Future research could focus on improving the system's resilience to handle a wider range of texts and languages and enhancing the evaluation methodologies

Innovation

This research stands out because of its innovative usage of Google's ChatGoogleGenerativeAI model for MCQ creation and OpenAI's GPT-3.5 for evaluation. While language models have been employed in different fields, their utilization in educational technology, notably for MCQ creation and evaluation, is relatively recent. The ChatGoogleGenerativeAI

model,part of the langchain_google_genai library, developed by Google, is recognized for its capacity to produce human-like prose, providing it with an effective tool for crafting MCQs. This research pioneers the application of these advanced models in MCQ production and evaluation, delivering a distinct addition to the area.

Result

GeneratedQuestions

After feeding our program an article about Machine Learning from Wikipedia the multiple-choice questions generated are as depicted in Fig.7:

	MCQ	Choices	Correct
1	Who coined the term 'machine learning'?	a-> Arthur Samuel || b-> Donald Hebb || c-> Walter Pitts || d-> Warren McCulloch	a
2	What was another term used for machine learning in the 1950s?	a-> Self-teaching computers || b-> Artificial intelligence || c-> Neural networks || d-> Computer science	a
3	When was the earliest machine learning model introduced?	a-> 1949 || b-> 1950 || c-> 1955 || d-> 1959	a
4	Whose work on neural structures influenced machine learning algorithms?	a-> Arthur Samuel || b-> Donald Hebb || c-> Walter Pitts || d-> Warren McCulloch	b
5	Which researchers proposed mathematical models of neural networks?	a-> Arthur Samuel and Donald Hebb || b-> Walter Pitts and Warren McCulloch || c-> Donald Hebb and Walter Pitts || d-> Arthur Samuel and Warren McCulloch	b

Fig. 7. Screenshot of Generate Questions

After being generated, the quiz is evaluated to determine its complexity and usefulness. This assessment entails comprehensive examination of the quiz is performed, including recommendations for any necessary tweaks or adjustments to guarantee maximum efficiency and pertinence. Fig. 8. is a screenshot of the complexity analysis

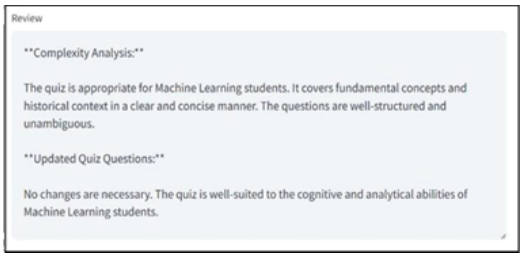

Fig. 8. Screenshot of MCQ Analysis/Review

Fig. 8. Screenshot of MCQ Analysis/Review Comparing Similarities Between Questions

After generating the questions, all the questions were compared to each other, and a similarity matrix was created to make sure no questions were repeated and asked about the same concept. Fig. 9 shows the similarity heatmap constructed using the similarity matrix. The similarity score can shed light on the variety and originality of the questions that are created. While a low similarity score may imply higher variety and quality, a high similarity score across questions may indicate repetition or a lack of variation in the created questions.

Comparing the similarity score against a baseline or human-generated questions can help evaluate the performance of the generative AI model. If the similarity score is comparable to or better than human-generated questions, it indicates that the model is effectively generating questions that are similar to those created by humans. High similarity scores between questions can identify redundant or repetitive questions in the generated dataset. This information can be valuable for refining

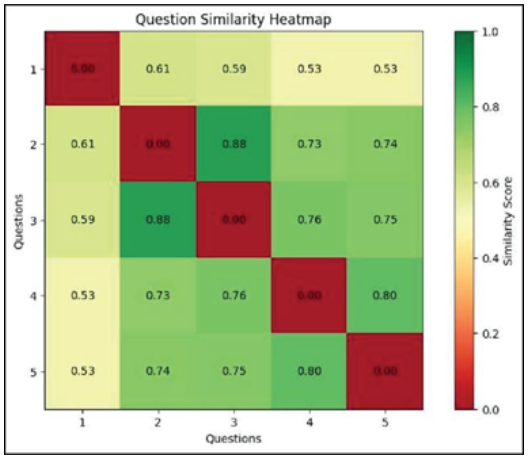

Fig. 9. Heatmap showing similarity score between questions

the generative AI model to produce more diverse and unique questions

Assessing the Difficulty of the Questions

Our evaluation of the automatically generated multiple-choice questions (MCQs) using generative AI encompassed an assessment of both option variation and question quality. We calculated the average similarity score between options for each generated question to discern the diversity and difficulty of the questions produced by the AI model and created a heatmap using the similarity scores (Fig. 10). Lower similarity scores among options may indicate greater variation or less challenging questions, whereas lower scores suggest closeness of the options to each other and potentially more challenging questions. The results of our analysis, presented in the figure below, provide a comprehensive overview of the average similarity between options of every question, offering valuable insights into the quality and diversity of the generated MCQs.

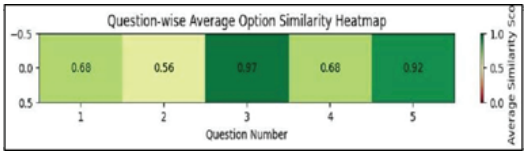

Fig. 10. Average option similarity heatmap

Conclusion

In conclusion, the MCQ generator project highlights the promise of advanced language models in educational technology. Utilizing Google's ChatGoogleGenerativeAI model in Python, the system constructs human-like MCQs and analyzes their quality. Emphasizing user-friendly interfaces with the streamlit framework, the project exhibits promising outcomes despite limitations. Future projects aim to solve these difficulties with a standalone application

Future Work

The current implementation of the MCQ generator has opportunity for development and growth. The future work on this project will focus on several major topics. The initial target for future development is to produce a standalone program that can be installed on a user's computer. This will allow the software to utilize the computer's resources more efficiently and give a better user experience. The standalone application will also eliminate any limits connected with web- based apps, such

as internet access concerns or server-side resource limitations.

Another area of focus is the incorporation of multiple Language Learning Models (LLMs) to cross-check the MCQs created. This will assure the accuracy of the created MCQs and deliver the most accurate answers to the user. The usage of numerous LLMs will also strengthen the robustness of the system. Providing performance and accuracy measurements to the user is another crucial part of future work. This involves giving an accuracy score relating to the correct answer and demonstrating how erroneous answers are closely related to the right answer. This can create a level of confusion in tests, making the examination more complex and sophisticated by modifying the accuracy index of the options. the subject of the questions to be generated. This could be based on the analysis of the input text and the user's requirements. In the future, the system could add adaptive learning techniques to increase the quality of the generated MCQs over time. This could involve learning from user input and past MCQs to refine the generating process. Future work could focus on extending support for multiple languages, making the system more accessible to users globally. While the current focus is on MCQs, the system might be modified to create additional types of questions, such as matching questions, and short answer questions.

References

1. Nagesh, C., Chaganti, K.R. , Chaganti, S. , Khaleelullah, S., Naresh, P. and Hussan, M. 2023. Leveraging Machine Learning based Ensemble Time Series Prediction Model for Rainfall Using SVM, KNN and Advanced ARIMA+ E-GARCH. International Journal on Recent and Innovation Trends in Computing and Communication. 11, 7s (Jul. 2023), 353–358. DOI:https://doi.org/10.17762/ijritcc.v11i7s.7010.
2. S. Khaleelullah, P. Marry, P. Naresh, P. Srilatha, G. Sirisha and C. Nagesh, "A Framework for Design and Development of Message sharing using Open-Source Software," 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), Erode, India, 2023, pp. 639-646, doi:10.1109/ICSCDS56580.2023.10104679.
3. Ramesh Kumar Ramaswamy, Pannangi Naresh, Chilamakuru Nagesh, Santhosh Kumar Balan, Multilevel thresholding technique with Archery Gold Rush Optimization and PCNN-based childhood medulloblastoma classification using microscopic images, Biomedical Signal Processing and Control, Volume 107, 2025,107801, ISSN 1746-8094, https://doi.org/10.1016/j.bspc.2025.107801.
4. K. R. Chaganti, B. N. Kumar, P. K. Gutta, S. L. Reddy Elicherla, C. Nagesh and K. Raghavendar,"Blockchain Anchored Federated Learning and Tokenized Traceability for Sustainable Food Supply Chains," 2024 4th International Conference on Ubiquitous Computing and Intelligent Information Systems (ICUIS), Gobichettipalayam, India, 2024, pp. 1532-1538, doi: 10.1109/ICUIS64676.2024.10866271.
5. Mohammed Inayathulla and Karthikeyan C, "Image Caption Generation using Deep Learning For Video Summarization Applications" International Journal of Advanced Computer Science and Applications(IJACSA), 15(1), 2024. http:// dx.doi.org/10.14569/IJACSA.2024.0150155.
6. Mohammed, I., Chalichalamala, S. (2015). TERA: A Test Effort Reduction Approach by Using Fault Prediction Models. In: Satapathy, S., Govardhan, A., Raju, K., Mandal, J. (eds) Emerging ICT for Bridging the Future - Proceedings of the 49th Annual Convention of the Computer Society of India (CSI) Volume 1. Advances in Intelligent Systems and Computing, vol 337. Springer, Cham. https://doi. org/10.1007/978-3-319-13728-5_25
7. Inayathulla, M., Karthikeyan, C. (2022). Supervised Deep Learning Approach for Generating Dynamic Summary of the Video. In: Suma, V., Baig, Z., Kolandapalayam Shanmugam, S., Lorenz, P. (eds) Inventive Systems and Control. Lecture Notes in Networks and Systems, vol 436. Springer, Singapore. https://doi.org/10.1007/978-981-19- 1012-8_18.
8. Mohammed and C. Karthikeyan, "Object detection in video summarization for video surveillance applications," Yugoslav Journal of Operations Research, vol. 35, no. 3, pp. 523–546, 2025. doi:10.2298/YJOR240615010I.
9. Inayathulla Mohammed and K. R. Rao, "Enhancing real-time violence detection in video surveillance using hybrid deep learning model," Journal of Web and User Applications, vol. 16, no. 1, 2025. doi:10.58346/JOWUA.2025.I1.021.
10. G. Raju, K. Sushmitha, M. Priyanka, E. Sai Leela, M. Vani Chowdary, and A. Yashwantha, "Real Time Hand Gesture Recognition Using CNN," Global Journal of Engineering Innovations & Interdisciplinary Research (GJEIIR), vol. 5, no. 2, 2025, doi: 10.33425/3066-1226.1101.
11. G. Raju, K. Sattar Hadiya, K. Penna Obulesu, M. P. Pavan Raj, and M. Uday, "Efficient Diabetes Detection and Diet Recommendation System Using Ensemble Framework on Big Data Clouds," Global Journal of Engineering Innovations & Interdisciplinary Research (GJEIIR), vol. 5, no. 2, 2025, doi: 10.33425/3066-1226.1108.
12. G. Raju, L. R. Buchupalli, A. Thudimela, K. Kodikondla, K. Palaku, and D. Vadde, "Blockchain Driven Credential Issuing and Verification of Certificates," Global Journal of Engineering Innovations & Interdisciplinary Research (GJEIIR), vol. 5, no. 2, 2025, doi: 10.33425/3066-1226.1083.