Journal of Ubiquitous Computing and Communication Technologies (ISSN: 2582-337X)
www.irojournals.com/jucct/

Automated Learning and Scheduling Assistant using LLM

Mohanraj K R.[1], Abinayasankar M.[2], Balaji G B.[3]

[1]Assistant Professor, [2,3]Student, Department of Information Technology, Velammal Engineering College, Tamilnadu, India

E-mail: [1]mohanraj@velammal.edu.in, [2]abinayasankar2004@gmail.com, [3]balajimsd098@gmail.com

Abstract

Large Language Models (LLMs) serve as the backbone of many AI applications, such as automatic content generation, virtual assistant and more. It is also used in automating the educational processes, such as scheduling the students' assessments and managing teachers' essential duties. The proposed study focuses on the design and development of an Automated Learning and Scheduling Assistant to facilitate the tasks like conducting unit test, managing internal assessment, and providing complete schedules of the students and the staff using LLM. The system is designed using the prompt engineering technique to improve the task automation efficiency. Retrieval-Augmented Generation (RAG) used helps in retrieving information and decision making, automating the test generation and the scheduling tasks. Data storage and retrieval are supported by the integration of the vector database. The primary objective of the proposed system is to enhance the educational process by automating the essential administrative and teaching functions, providing a scalable solution for the modern learning environment.

Keywords: LLM, RAG, Prompt Engineering, Education Process, Automation.

1. Introduction

The rapid development in Artificial Intelligence (AI) has made it pivotal in modern educational frameworks, as it provides unparalleled support in personalized learning, administrative efficiency and teacher productivity. One such AI advancement is LLMs, which

DOI: https://doi.org/10.36548/jucct.2024.3.006
Received: 02.08.2024, received in revised form: 21.08.2024, accepted: 08.09.2024, published: 20.09.2024

can process and understand large amount of text, transforming the way information is delivered and absorbed in the education. The proposed Automated learning and Scheduling Assistant represents an advanced automated technique of question generation and other assessments using RAG and LLM concepts. The system, specifically uses the LLM Gemini, which is activated by prompts to improve its efficiency in automatically assessing the tests and generating report based on the test results. The user interface is designed to automate various processes using LLM and machine learning concepts. It uses FastAPI with a React front end to provide a user-friendly interface that supports auto monitored tests and assistive AI functionalities. The system also uses the test results to facilitate the event generation, enabling automatic task scheduling.

Artificial Intelligence (AI) is revolutionizing task scheduling by optimizing the processes and enhancing efficiency. AI-powered scheduling assistants use Natural Language Processing (NLP) to understand and respond to human queries, prioritizing tasks and making informed decisions [1-4]. These systems improve user experience by accurately interpreting variations in language. Retrieval-Augmented Generation (RAG) enhances text generation by combining retrieval-based methods with generative models, producing contextually rich and accurate responses [5]. NLP-based reinforcement learning improves the system's understanding of natural language commands and context, allowing it to refine decision-making capabilities [6]. RAG-guided NLP techniques enhance accuracy by optimizing task prioritization and scheduling processes [7]. A reinforcement learning approach using NLP embeddings improves contextual understanding and decision-making capabilities [8]. Deep reinforcement learning combined with NLP-driven rewards enhances decision-making capabilities and overall performance [9-10].

The proposed study focuses on developing a user-friendly interface integrated with advanced AI tools to automate the administrative and the teacher functions, thereby enhancing the education process.

2. Related Works

This research introduces DocChat, a novel information retrieval technique for chat-bot engines that can directly determine the relevance between utterances and responses by

using unstructured texts to reply to utterances instead of question-response (Q-R) pairs [11]. As Organizations are increasingly adopting conversational agents (CAs) for personalized services and information, many lack knowledge on evaluating and improving their quality. This design science research aims to fill this gap by aggregating literature and practice insights, providing a blueprint for CA evaluation and systematic improvement [12]. Retrieval-Augmented Generation (RAG) improves Large Language Model (LLM) output by providing context to input. In this study, Case-Based Reasoning (CBR) is integrated into RAG for LLMs, enhancing queries with contextually relevant cases. Evaluation of CBR-RAG shows that CBR's case reuse improves the quality of generated answers, enhancing the task of legal question-answering. The results show that combining RAG with rules-based AI, enhances data processing through logic-based rules, improves LLM accuracy while maintaining flexibility, making it a promising approach for production systems [13].

Researchers are also exploring reinforcement learning (RL) algorithms for natural language processing (NLP) tasks, particularly in conversational systems. This review examines the state of RL methods, discussing their potential for solving problems, analyzing advantages and limitations, and highlighting promising research directions in NLP [14]. To enhance spoken language understanding (SLU), this research suggests using a retrieval augmented generation (RAG) method. It introduces a prompt attention technique and uses a pretrained automatic speech recognition model encoder for speech retrieval, phrased as prompts for SLU decoder. Experiments demonstrate that this method performs better in intent prediction than traditional models [15].

The proposed study focuses on the development of an Automated Learning and Scheduling Assistant to facilitate tasks like unit tests, internal assessments, and student and staff schedules. The system uses prompt engineering techniques, pretrained transformers, Retrieval-Augmented Generation (RAG), vector database integration, and reinforcement learning to enhance the educational process and provide a scalable solution.

3. Proposed System

The “Automated Learning and Scheduling Assistant” operates through various integrated process. The users specify the assessment details, such as course topic, difficulty

level, and type of assessment required. The parameters defined by the users, initiate the Retrieval-Augmented Generation (RAG), that uses the historical data about the previous test and the students' performance available in the database. This data from RAG is combined with the prompt engineered technique to generate the prompts that is used by the LLM to generate questions for the assessment. In the same way the inputs on the course topics, staff and resources availability, initiates the RAG to use the historical data on the previous schedules, the staff work load, their performance in particular subject, and their resource availability are used along with the prompt engineering technique and LLM to generate the staff schedule. The Large Language Models (LLM) ensures that the questions and schedule reflect the goals. The Figure.1 below shows the overall workflow of the proposed.

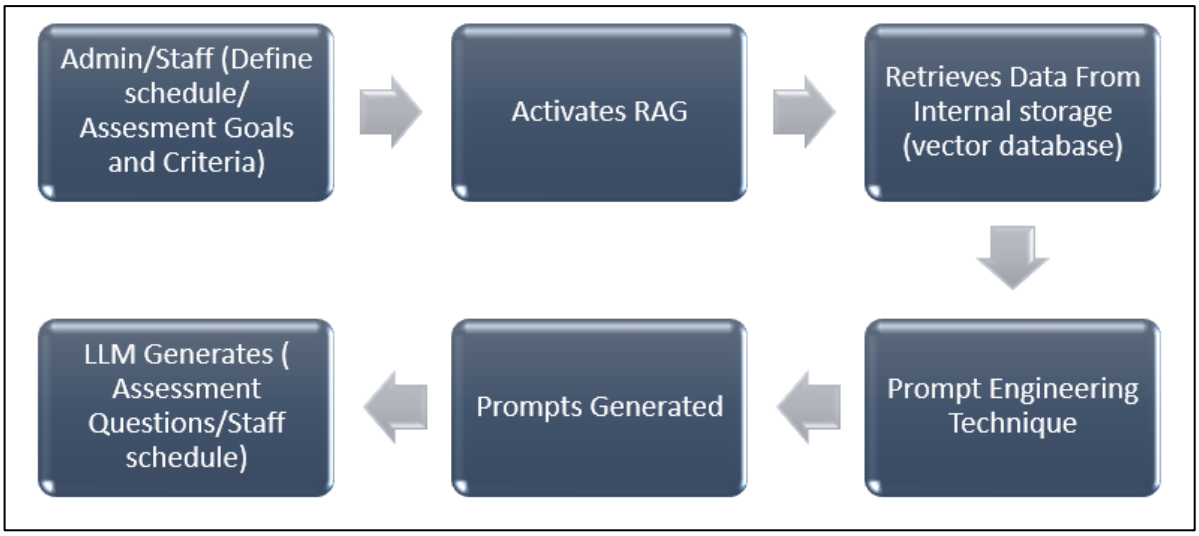

Figure 1. Workflow of the Proposed System

The user interface developed for automatic learning and scheduling assistant has two major modules they are the internal assessment module and the scheduling module.

The Internal Assessment module generates assessments for students based on the courses they have opted for, using input parameters defined by the staff, such as course topic, difficulty level, type of assessment, etc. These inputs activate the RAG that has a built-in mechanism to understand the input and generate queries to find and retrieve relevant data from the vector database that serves a structured repository and holds data, such as question formats, past exam papers, students' performance metrics, etc. The retrieved data is integrated into the response generation process. This retrieved data is used by the prompt engineering

techniques to generate prompts. Template-based prompts are initially generated based on user input, which are further customized using contextual adaptation and augmented with data retrieved by RAG. The final prompts are utilized by the Large Language Model (LLM) to generate and verify the assessment questions. The final prompts are used by the LLM to generate and verify the outputs. Similar steps are followed in staff scheduling according to the input parameters defined by the administrators, such as course topics, staff name, resources available etc.

The technical requirements for the system include the use of Google Gemini as the Large Language Model (LLM) to manage text generation and prompt engineering. The system incorporates Retrieval-Augmented Generation (RAG) technology to combine retrieval-based methods with generative models for enhanced content generation. Additionally, FAISS is utilized as the vector database for efficient storage and retrieval of vector embeddings. The user interface is built with React for the frontend, while the backend is developed using FastAPI, ensuring a seamless integration between the various components of the system. The Figure.2 below shows the overview of user interface.

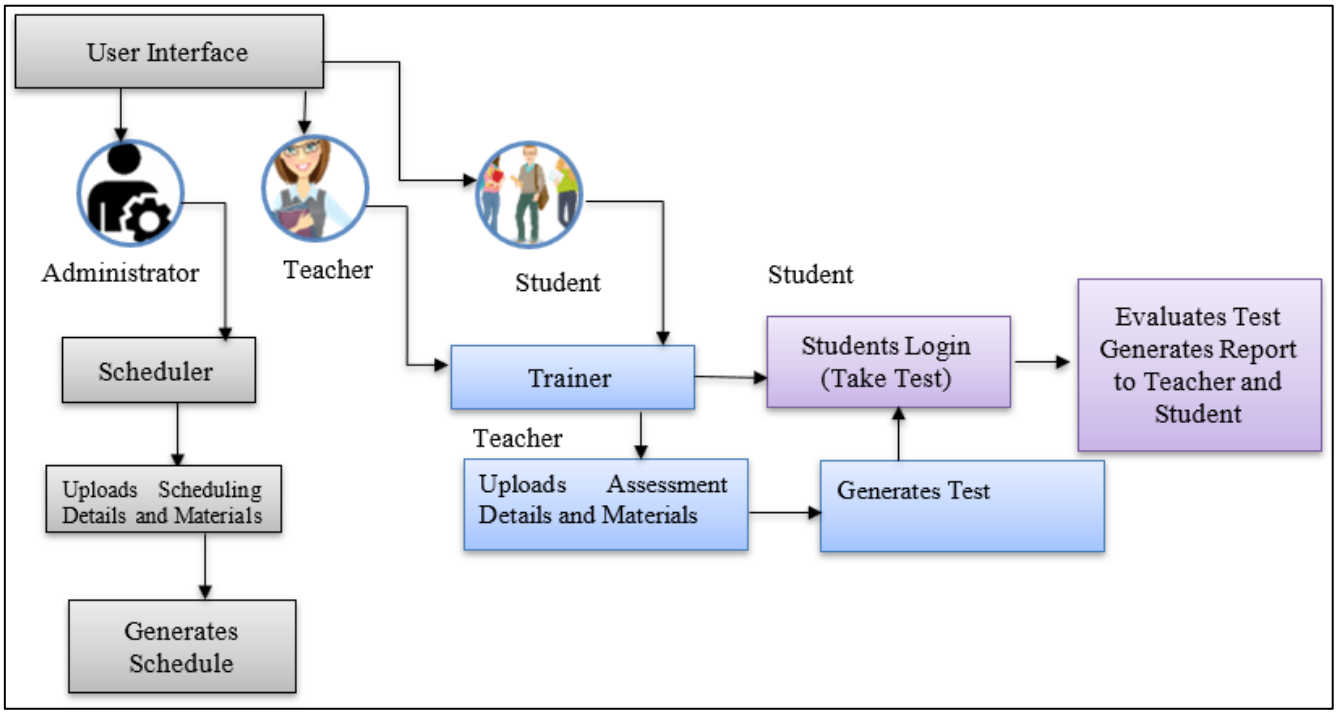

Figure 2. Overview of the User Interface

As shown in figure above, the teacher or the trainer uploads the documents required for test generation in PDF formats and specifies the necessary input to conduct the test. The uploaded PDF is converted into vectors and stored in the vector database. The relevant content is then retrieved by the RAG from the vector database using similarity search based on the specified input and combined with the prompt engineering techniques to generate question using LLM. The students can log in using the student credentials and take the test. Similarly, the administrator uploads the document and provides the specific inputs required for generating the staff schedule.

4. Results

The hardware requirements of the system include an Intel i5 processor with 16 GB of RAM and a 256 GB SSD. The system was developed on Windows OS. The frontend was developed using JavaScript, while Python was used as the core language for the backend, along with PyTorch libraries and MySQL.

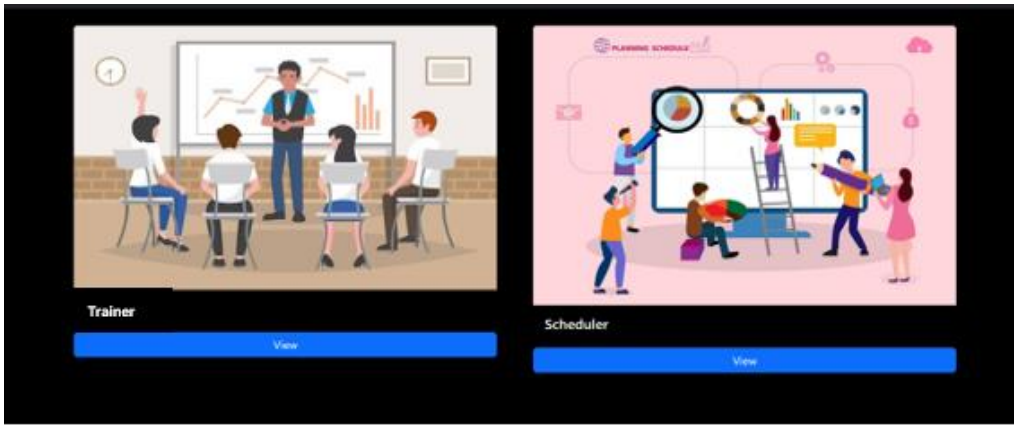

Figure 3. Home Page

Figure 3 displays the home page of the user interface, featuring two major modules: The Trainer and the Scheduler.

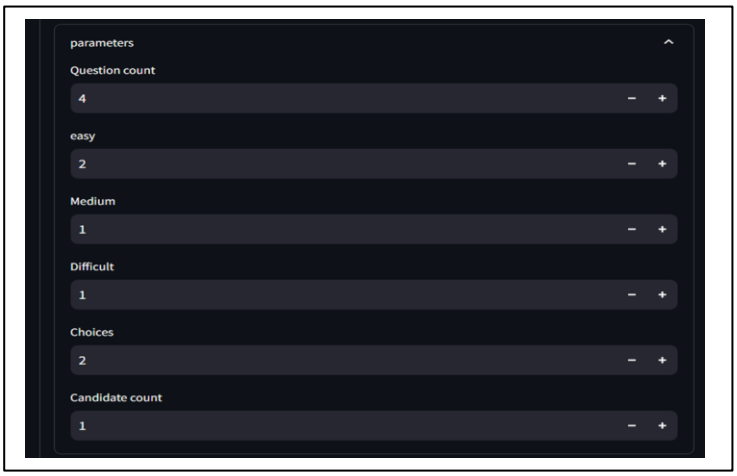

Figure 4. Parameter Specification Page

Figure 4 depicts the specifications for the parameters that the test and assessment model should adhere to. It shows how this data is provided to the LLM when needed to process and generate efficient outputs.

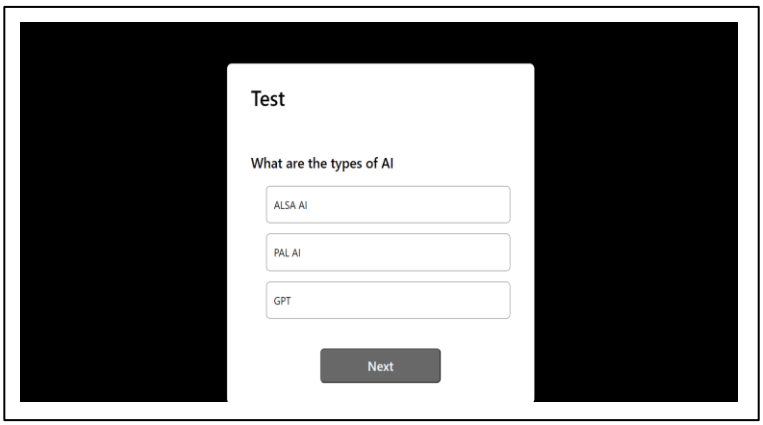

Figure 5. Quiz Page

Figure 5 illustrates the test content generated by the Gemini LLM, using fine-tuned parameters to optimize performance. The Top_p parameter controls the randomness of word selection, with values ranging from 0 to 1. A higher value (>0.5) produces more diverse content, and for this system, a value of 0.7 is selected to strike a balance between creativity and coherence. The Top_k parameter limits the number of potential next words the model can consider, ranging from 0 to 10. A value of 4 is used to ensure more precise and relevant content generation. Additionally, the Temperature parameter affects the creativity of the generated content, with higher values (>5) resulting in more varied and innovative outputs.

For this system, a temperature value of 8 is chosen to maintain a balance between creativity and relevance to the content extracted from the provided PDFs.

From the students' perspective, while they are taking the test, the system allocates a Langchain buffer memory. This memory stores the data during the test and logs it into the database once the test is completed. Meanwhile, the LLM processes the student's answers in real time, and performance data is stored. Q-learning analyzes this data, updating its policy to optimize question difficulty and focus based on the student's performance. The LLM then uses this updated policy to generate additional questions tailored to the student's needs, providing an adaptive and personalized testing experience.

The development of the user interface is still under process and aims for a successful implementation of the website to make it available for academic use in future. Though the system seems to be highly convenient for the teachers as well as the administrators by automating the process and reducing the human intervention. The system still faces certain challenges due its high dependency on historical data. Inadequate or biased data can lead to inaccurate question generation as well as scheduling, additionally the differences in student performance could affect the system ability to personalize the assessment effectively. This makes the user feedback and robust testing essential to enhance the overall effectiveness and applicability of the automatic learning and the scheduling assistant using LLM technology.

5. Conclusion and Future Works

The Automated Learning Scheduling Assistant represents a significant advancement in scheduling technology, combining sophisticated machine learning algorithms, natural language processing, and user-friendly interfaces to streamline scheduling tasks. By offering personalized scheduling recommendations, automatic conflict resolution, and adaptive schedule adjustments, it enhances productivity and empowers users to manage their time more effectively. In future, the system could benefit from enhancements such as improved natural language understanding, advanced scheduling algorithms, and expanded integration with external services. Incorporating user feedback and advanced analytics will further refine its functionality, ultimately revolutionizing how individuals and organizations approach scheduling and time management in the digital age.

References

[1] Chen, Lijia, Pingping Chen, and Zhijian Lin. "Artificial intelligence in education: A review." IEEE Access 8 (2020): 75264-75278.

[2] Lalwani, Tarun, Shashank Bhalotia, Ashish Pal, Vasundhara Rathod, and Shreya Bisen. "Implementation of a Chatbot System using AI and NLP." International Journal of Innovative Research in Computer Science & Technology (IJIRCST) Volume-6, Issue-3 (2018).26-30.

[3] Karthick, S., R. John Victor, S. Manikandan, and Bhargavi Goswami. "Professional chat application based on natural language processing." In 2018 IEEE International Conference on Current Trends in Advanced Computing (ICCTAC), Bangalore, India IEEE, 2018. 1-4.

[4] Qin, Youming, Wei Xu, Adrian Lee, and Fu Zhang. "Gemini: A compact yet efficient bi-copter uav for indoor applications." IEEE Robotics and Automation Letters 5, no. 2 (2020): 3213-3220.

[5] Chen, Wei, Jincai Chen, Fuhao Zou, Yuan-Fang Li, Ping Lu, Qiang Wang, and Wei Zhao. "Vector and line quantization for billion-scale similarity search on GPUs." Future Generation Computer Systems 99 (2019): 295-307.

[6] Tapsai, Chalermpol. "Information processing and retrieval from CSV file by natural language." In 2018 IEEE 3rd International Conference on Communication and Information Systems (ICCIS), Singapore. IEEE, 2018. 212-216.

[7] Liang, Yuanyuan, Jianing Wang, Hanlun Zhu, Lei Wang, Weining Qian, and Yunshi Lan. "Prompting large language models with chain-of-thought for few-shot knowledge base question generation." arXiv preprint arXiv:2310.08395 (2023).

[8] Sonkar, Shashank, Andrew E. Waters, and Richard G. Baraniuk. "Attention word embedding." arXiv preprint arXiv:2006.00988 (2020).

[9] Madotto, Andrea, Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. "Few-shot bot: Prompt-based learning for dialogue systems." arXiv preprint arXiv:2110.08118 (2021).

[10] Abbasian, Mahyar, Iman Azimi, Amir M. Rahmani, and Ramesh Jain. "Conversational health agents: A personalized llm-powered agent framework." arXiv preprint arXiv:2310.02374 (2023).

[11] Yan, Zhao, Nan Duan, Junwei Bao, Peng Chen, Ming Zhou, Zhoujun Li, and Jianshe Zhou. "Docchat: An information retrieval approach for chatbot engines using unstructured documents." In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany. 2016. 516-525.

[12] Lewandowski, Tom, Emir Kučević, Stephan Leible, Mathis Poser, and Tilo Böhmann. "Enhancing conversational agents for successful operation: A multi-perspective evaluation approach for continuous improvement." Electronic Markets 33, no. 1 (2023): 39.

[13] Wiratunga, Nirmalie, Ramitha Abeyratne, Lasal Jayawardena, Kyle Martin, Stewart Massie, Ikechukwu Nkisi-Orji, Ruvan Weerasinghe, Anne Liret, and Bruno Fleisch. "CBR-RAG: case-based reasoning for retrieval augmented generation in LLMs for legal question answering." In International Conference on Case-Based Reasoning, Cham: Springer Nature Switzerland, 2024. 445-460.

[14] Uc-Cetina, Victor, Nicolás Navarro-Guerrero, Anabel Martin-Gonzalez, Cornelius Weber, and Stefan Wermter. "Survey on reinforcement learning for language processing." Artificial Intelligence Review 56, no. 2 (2023): 1543-1575.

[15] Yang, Hao, Min Zhang, Daimeng Wei, and Jiaxin Guo. "Srag: speech retrieval augmented generation for spoken language understanding." In 2024 IEEE 2nd International Conference on Control, Electronics and Computer Technology (ICCECT), Jilin, China. IEEE, 2024. 370-374.