IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Leveraging LangChain LLM Framework And LangGraph For Building A Dynamic Education Application With Context-Aware Conversational Agents

Dr .Raghu Kumar K S, Aftab J, Javeria Nazeer Ahmed, Mohammad Saad Khan, Syed Ishtiaq Ahmed

Associate Professor, Student, Student, Student, Student

Department Of Artificial Intelligence and Machine Learning

Vijaya Vittala Institute of Technology, Bengaluru, India

Abstract : This paper presents a dynamic education application developed using the LangChain LLM framework and LangGraph agent architecture to enhance personalized learning and educational management. The system integrates a context-aware conversational agent, built with LangGraph, enabling users to interact with educational content, manage resources, and receive tailored assistance. Core functionalities include automatic quiz generation based on user-provided titles or topics, class-wise report generation from SQL database records, and AI-powered report writing that leverages large language models for insights and recommendations. A centralized dashboard visualizes student performance data, supporting educators in tracking progress and identifying areas for improvement. Additionally, the platform offers a student management system for maintaining student records and managing learning resources such as quizzes and PDFs. By combining conversational AI with database-driven insights, the application provides an interactive, data-informed, and agent-based solution for modern educational environments.

Keywords - Langchain , LangGraph, Conversational Agent, Matplotlib, Dynamic Education System, Quiz Generation, Student Performance Dashboard

I INTRODUCTION

The advancement of large language models (LLMs) and AI frameworks has enabled the development of intelligent, personalized education systems. This project presents a dynamic education application using the LangChain framework and LangGraph agent architecture to integrate conversational AI with database-driven insights. The system features a context-aware conversational agent, built with LangGraph, that allows users to interact with educational content, generate quizzes dynamically based on titles or topics, and manage learning resources. It connects to a SQL database to fetch student data and uses Matplotlib to visualize performance metrics through an interactive dashboard. These visual insights support automatic report generation, providing personalized feedback and recommendations for improvement. Additionally, the application includes a student management system to maintain records and organize resources such as quiz PDFs. By combining LangChain, LangGraph, and data visualization tools, the system offers an interactive, data-driven, and scalable solution for modern educational environments.

II OBJECTIVE

The objective of this project is to develop a dynamic education application using LangChain and LangGraph that integrates a context-aware conversational agent with a database-driven backend to enhance teaching and learning processes. The system aims to automate quiz generation based on input topics, generate class-wise reports from SQL database records, and provide personalized recommendations for students based on performance data. It includes a dashboard for visualizing student performance using Matplotlib, a student management system for organizing student records, and tools to manage educational resources such as quiz PDFs. The overarching goal is to create an interactive, intelligent, and scalable platform that combines AI-powered conversation, data visualization, and educational resource management to support both educators and learners.

III LITERATURE SURVEY

Recent advancements in large language models (LLMs) and conversational AI have enabled the development of intelligent educational tools, combining natural language processing, data visualization, and chatbot technologies. Researchers have explored how LLMs can support interactive learning, personalized assistance, and effective data interpretation in educational contexts.

- Santhosh Ram and Muthumanikandan (2024) developed *Visistant*, a chatbot that translates natural language queries into visual data using Gemini LLMs, making data visualization accessible through conversation.
- Kumar et al. (2024) created an interactive academic assistant leveraging LLMs to provide personalized learning support and resource generation for students and teachers.
- Chaubey et al. (2024) performed a comparative analysis of RAG, fine-tuning, and prompt engineering to evaluate and improve chatbot performance in educational applications.

IV PROPOSED SYSTEM

The proposed system is a dynamic educational portal that integrates multiple AI-powered features into a unified platform. It includes modules for quiz generation, report analysis, student management, and an intelligent chat agent, all connected through a backend that leverages large language models (LLMs) and LangGraph-based agents. The quiz generation module accepts a topic input, uses an LLM to create a set of questions, saves them as a PDF, and updates the metadata for future access. The report analysis module retrieves student data from the database, analyses performance metrics, and generates class-wise visual reports. The student management system provides CRUD (Create, Read, Update, Delete) operations for managing student records and ensures seamless data manipulation with direct database interaction. The chat agent functions as a context-aware conversational interface; it uses LangGraph to determine if user queries require general responses or database access. For database-related queries, it generates and executes SQL queries dynamically, providing results directly through the chat. All modules interact with core resources such as user data, quiz metadata, PDFs, and the database, enabling a comprehensive, intelligent educational experience through a centralized dashboard.

Proposed System Diagram

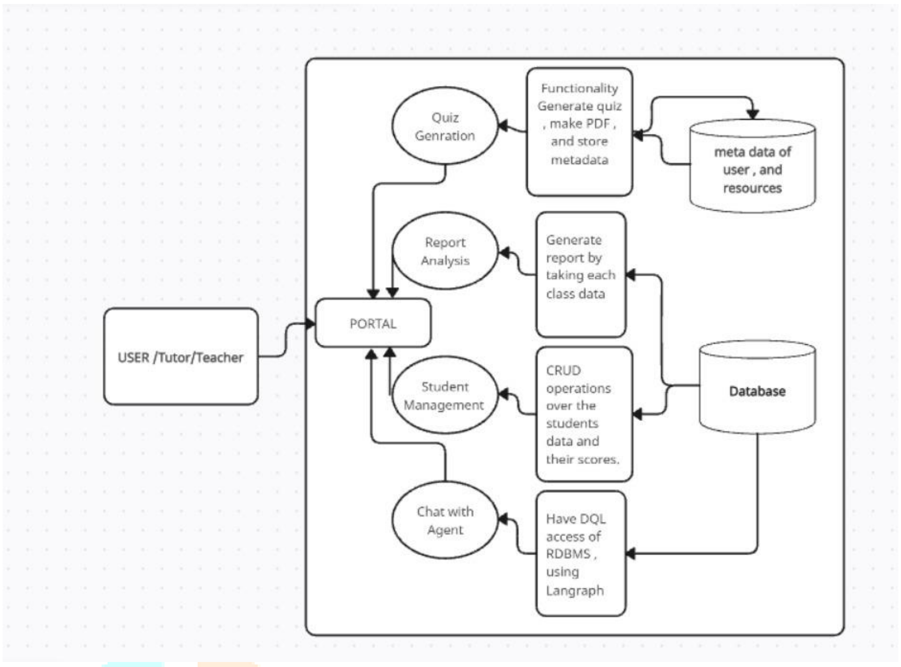

Quiz Genration
Functionality Generate quiz , make PDF , and store metadata
meta data of user , and resources
Report Analysis
Generate report by taking each class data
USER /Tutor/Teacher
PORTAL
Student Management
CRUD operations over the students data and their scores.
Database
Chat with Agent
Have DQL access of RDBMS , using Langraph

Fig. 1. Proposed System

Quiz Generation Flow

In the quiz generation module, the user provides a topic or title through the portal interface. This input is passed to a backend powered by a large language model (LLM), which generates a set of quiz questions and answers relevant to the topic. The generated quiz is automatically formatted and saved as a PDF file. Alongside the PDF, metadata such as the quiz title, date, and number of questions is stored in a database for future retrieval and management. This allows users to download, view, or reuse quizzes efficiently from the portal.

Report Analysis Flow

The report analysis module collects student performance data from the backend database, segmented by class or group. Using data analysis tools and libraries like Matplotlib, it processes the data to create visual reports, such as bar charts or performance summaries. These reports are dynamically displayed on the dashboard, providing educators with insights into class-level and individual student performance. Based on the analysis, the system can also generate personalized recommendations or suggestions to help improve learning outcomes.

Student Management System Flow

The student management system enables CRUD operations—allowing administrators to create, read, update, or delete student records directly through the portal. Each action triggers backend operations that interact with the database in real-time, ensuring up-to-date records. Users can view student details, add new entries, edit information, or remove outdated records, all within a user-friendly interface. This module ensures seamless management of student data while maintaining data integrity and easy access.

Chat with Agent Flow

The chat module uses a LangGraph-based agent to enable intelligent, context-aware conversations. When a user asks a question, the agent first checks whether the query is a general question or one that requires database access. If it's a database-related query (e.g., "What is the average score of Class A?"), the agent generates an appropriate SQL query, executes it on the database, and returns the result through the chat interface. For general queries, it uses the LLM directly to generate conversational responses. This approach allows the agent to handle both knowledge-based and data-driven interactions intelligently.

V SOFTWARE REQUIREMENTS AND USED TECHNOLOGIES

A - Tools and Technologies

- Programming languages: Python3
- Large Language Model (LLM): grokCloud Llama3-70B-8192
- Frameworks and Libraries: Langchain, LangGraph, Matplotlib, SQLite, JSON, fpdf
- Graphical User Interface (GUI) Tools: Tkinter, PyQT, Streamlit

B – Hardware Requirements

- Processor – Intel Core i5 (minimum)
- RAM – 8 GB (minimum)
- Storage – 512 GB
- Operating System – Windows 10/11, Linux(Ubuntu 20.04+),or macOS

VI FLOW OF SYSTEM

1. User inputs a topic, and the LLM generates a quiz, saves it as a PDF, and stores metadata for future use.
2. The system fetches student data from the database, analyses it, and generates visual performance reports using Matplotlib.
3. Administrators can perform CRUD operations on student records, which are updated in real-time in the backend database.
4. The LangGraph agent processes queries, generates SQL queries for database-related questions, and provides conversational answers for others.
5. The system manages resources like quiz PDFs, metadata, and student performance data, ensuring seamless access and manipulation.

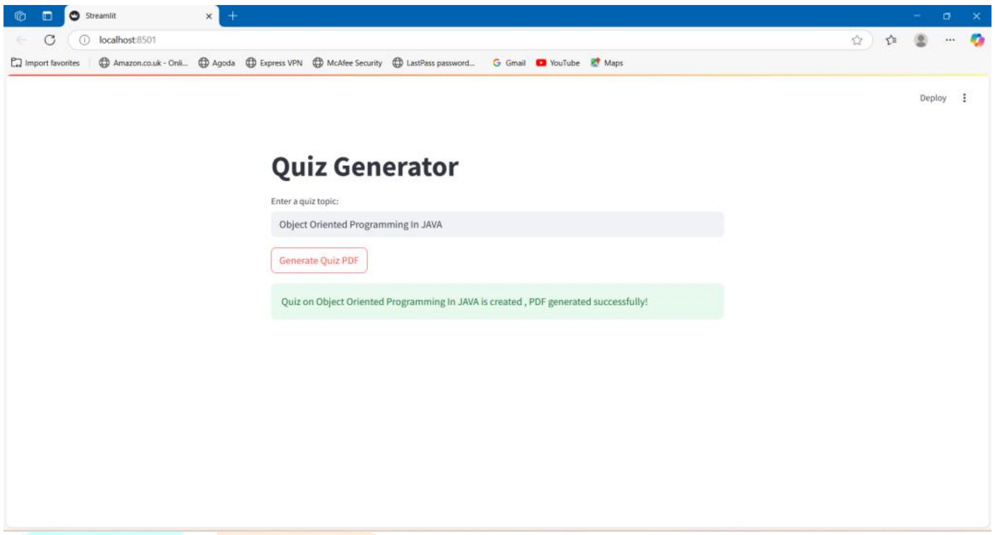
VII RESULT

The system successfully integrates multiple features aimed at enhancing the educational experience through automation and data-driven insights. The Quiz Generation module efficiently creates quizzes based on user input and stores them as PDFs, complete with metadata for easy access. The Report Analysis module visualizes student performance data, offering actionable insights via charts and summaries. The Student Management System allows for seamless CRUD operations, ensuring efficient management of student records. The Chat with Agent feature utilizes a LangGraph-powered agent to handle both general queries and database-driven questions, offering real-time responses. Lastly, the Resource Management component effectively organizes and stores quizzes and related data, ensuring smooth accessibility and manipulation for users. These results demonstrate the system's ability to support dynamic and efficient educational management through intelligent automation.

1) Quiz Generation
The quiz generation module creates quizzes based on user input, saves them as PDFs, and stores metadata for easy access.

Fig 2 Quiz Generation Functionality

2) Chat with Database



Quiz Generator

Enter a quiz topic:

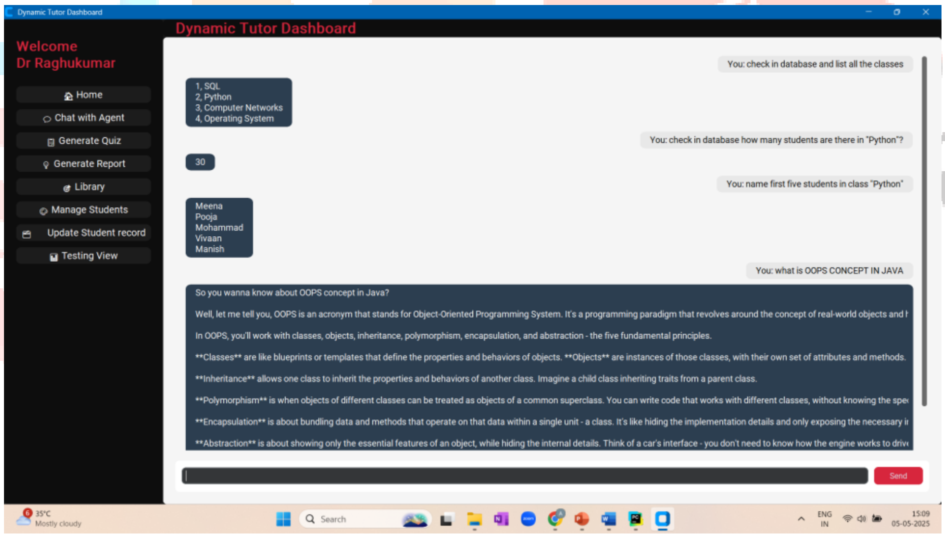
Object Oriented Programming In JAVA

Generate Quiz PDF

Quiz on Object Oriented Programming In JAVA is created , PDF generated successfully!

The chat module responds to user queries, handling both database-related questions (e.g., fetching student scores) and general inquiries.

Fig 3 Chat with database



Dynamic Tutor Dashboard

Welcome
Dr Raghukumar

You: check in database and list all the classes

1, SQL
2, Python
3, Computer Networks
4, Operating System

You: check in database how many students are there in "Python"?

30

You: name first five students in class "Python"

Meena
Pooja
Mohammad
Vivaan
Manish

You: what is OOPS CONCEPT IN JAVA

So you wanna know about OOPS concept in Java?

Well, let me tell you, OOPS is an acronym that stands for Object-Oriented Programming System. It's a programming paradigm that revolves around the concept of real-world objects and I

In OOPS, you'll work with classes, objects, inheritance, polymorphism, encapsulation, and abstraction - the five fundamental principles.

Classes are like blueprints or templates that define the properties and behaviors of objects. **Objects** are instances of those classes, with their own set of attributes and methods.

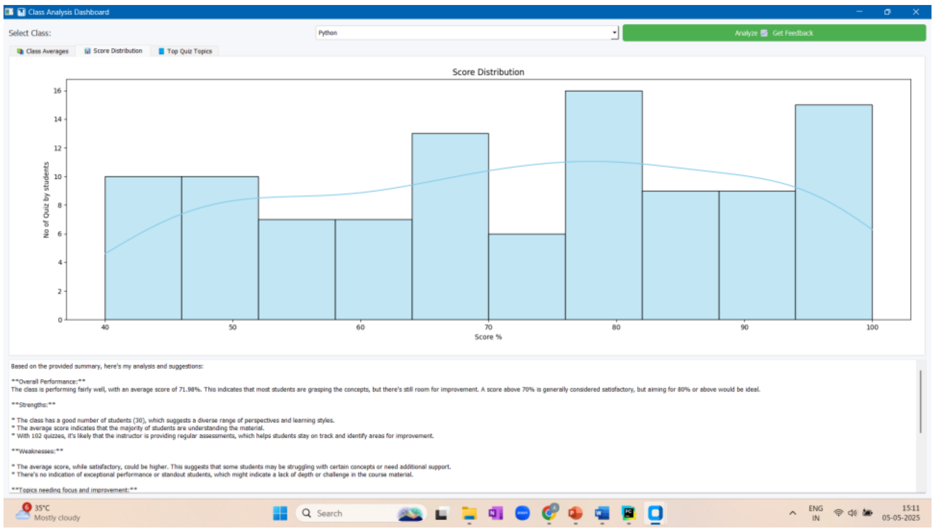
Inheritance allows one class to inherit the properties and behaviors of another class. Imagine a child class inheriting traits from a parent class.

Polymorphism is when objects of different classes can be treated as objects of a common superclass. You can write code that works with different classes, without knowing the spe

Encapsulation is about bundling data and methods that operate on that data within a single unit - a class. It's like hiding the implementation details and only exposing the necessary i

Abstraction is about showing only the essential features of an object, while hiding the internal details. Think of a car's interface - you don't need to know how the engine works to driv

3) Report Analysis Results
The report analysis module generates visual insights, offering performance trends and summaries of student data.

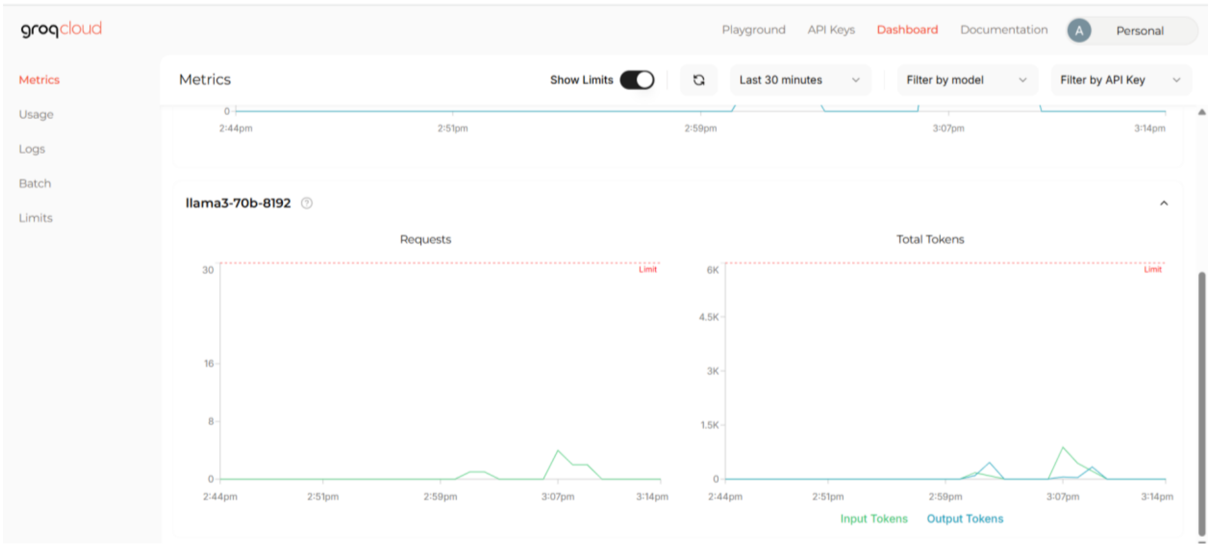


```
Based on the provided summary, here's my analysis and suggestions:

**Overall Performance:**
The class is performing fairly well, with an average score of 71.98%. This indicates that most students are grasping the concepts, but there's still room for improvement. A score above 70% is generally considered satisfactory, but aiming for 80% or above would be ideal.

**Strengths:**

* The class has a good number of students (30), which suggests a diverse range of perspectives and learning styles.
* The average score indicates that the majority of students are understanding the material.
* With 102 quizzes, it's likely that the instructor is providing regular assessments, which helps students stay on track and identify areas for improvement.

**Weaknesses:**

* The average score, while satisfactory, could be higher. This suggests that some students may be struggling with certain concepts or need additional support.
* There's no indication of exceptional performance or standout students, which might indicate a lack of depth or challenge in the course material.

**Topics needing focus and improvement:**
```

Fig 4 : Report Analysis Results

4) GrokCloud LLM Token usages

Fig 5: LLM Token Usages



VIII CONCLUSION AND FUTURE SCOPE

The proposed educational management system successfully integrates quiz generation, report analysis, student management, intelligent chat assistance, and resource handling into a unified platform. By leveraging large language models, LangChain, and LangGraph frameworks, the system automates quiz creation, provides insightful performance reports, and facilitates seamless data management through a user-friendly interface. The inclusion of an AI-powered chat agent further enhances user interaction by handling both conversational and database-related queries intelligently.

In the future, the system can be expanded to support multilingual quiz generation, adaptive learning paths based on student performance, integration with cloud databases for scalability, and mobile platform support for greater accessibility. Additionally, incorporating advanced analytics and AI-driven recommendations can further personalize the learning experience for students and educators.

IX REFERENCES

[1] G. K. Santhosh Ram and V. Muthumanikandan, "Visistant: A Conversational Chatbot for Natural Language to Visualizations with Gemini Large Language Models," *IEEE Access*, vol. 12, pp. 1–10, 2024. doi:10.1109/ACCESS.2024.3465541.

[2] P. Kumar, M. Haresh, and V. Hayagreevan, "Development of Interactive Assistance for Academic Preparation Using Large Language Models," *Proceedings of the International Conference on Advances in Computing*, pp. 150–155, 2024.

[3] H. K. Chaubey, G. Tripathi, S. K. Gopalaiyengar, and R. Ranjan, "Comparative Analysis of RAG, Fine-Tuning, and Prompt Engineering in Chatbot Development," *2024 International Conference on Future Technologies for Smart Society (ICFTSS)*, pp. 220–225, 2024.