INTEL-UNNATI AI&ML JOURNAL
Number 2 (2025), pp. 1–11

Code:IUIT-2025

MACHINE LEARNING

EduGenie : AI Powered Interactive Learning Assistant for Classrooms

Anjitha Anil, Ashwini Anil , and Anjali Thomas

Saintgits Group of Institutions, Kottayam, Kerala

Abstract: Traditional education systems often fail to accommodate the diverse learning styles and paces of students, resulting in gaps in understanding and engagement. To address this, we present EduGenie, an AI-powered interactive learning assistant designed to support self-paced, multimodal education. EduGenie enables students to input a topic via text or voice, upon which it generates concise, easy-to-understand notes using a Large Language Model (FLAN-T5), optimized for performance with Intel's OpenVINO Toolkit. In addition, it creates personalized quizzes based on the generated notes to reinforce learning. Students receive instant feedback and graphical score visualization, which improves engagement and comprehension. In addition, the system includes a Gemini API-powered 7-day study planner and PDF download support for offline study. The project is developed using a full-stack MERN (MongoDB, Express.js, React.js, Node.js) architecture. Our goal is to bridge the gap between AI and personalized education, providing an inclusive platform that caters to varied cognitive preferences, including learners based on auditory, visual, and text. Thus, EduGenie represents a scalable, intelligent learning solution for the future of education.

Keywords: AI-powered learning, generative AI, LLM, OpenVINO, quiz generation, multimodal learning, EduGenie, FLAN-T5, Gemini API, interactive education

1 Introduction

In the digital era, traditional classroom environments often struggle to meet the personalized learning needs of students. Learners differ in their pace, style, and cognitive preference: some may prefer auditory learning, others textual or visual content. However, conventional education systems largely follow a one-size-fits-all approach, limiting student engagement and retention. With the emergence of generative AI and Large Language Models (LLMs), there is a growing opportunity to reshape how learning materials are created and delivered.

To address this, we propose **EduGenie**: an AI-powered interactive learning assistant that helps students understand topics through concise AI-generated notes and quizzes. Students can input a topic via voice or text, and the system responds by generating simplified, easy-to-understand notes using the `Google/flan-t5-large` model. These notes are also used to create multiple choice questions, allowing learners to test their knowledge immediately through an interactive quiz interface. Real-time feedback, scoring, and graphical result summaries make the experience engaging and insightful.

Our system supports multimodal learning and is optimized using Intel's OpenVINO toolkit to ensure fast and efficient inference, even on edge devices. We also integrate a 7-day study planner using the Gemini API and offer a PDF export option for generated notes. This paper outlines the architecture, implementation, and potential of EduGenie to revolutionize personalized and inclusive digital learning.

2 Libraries Used

In the project, for various tasks, the following packages and libraries are used.

```
React.js
Axios
React Router
SpeechRecognition
jsPDF
Express.js
Transformers (Hugging Face)
OpenVINO Toolkit
Gemini API
Python
```

3 Methodology

The development of the EduGenie project follows a modular and systematic approach that integrates front-end, back-end, and AI components. The major stages involved in the implementation are:

User Input: The user provides a topic either by typing or through voice input using Web Speech API via the SpeechRecognition library.

Voice Recognition: If the user opts for voice, speech-to-text conversion is handled using React Speech Recognition to extract the topic in text form.

AI Note Generation: The extracted topic is sent to the backend where the Large Language Model `google/flan-t5-large`, optimized using Intel's OpenVINO Toolkit, is used to generate concise and structured notes.

Quiz Generation: The same topic is forwarded to a separate backend route, where quiz questions are generated dynamically using an LLM (such as GPT or Gemini API) and formatted into multiple-choice questions.

Notes Display and PDF Export: The generated notes are displayed on the frontend with an option to download them as a PDF using jsPDF.

Quiz Interaction: The user takes the quiz interactively. The application records the user's choices, calculates the score, and provides instant feedback.

Study Planner Generation: Additionally, a 7-day AI-generated study planner is offered using Gemini API integration to assist the learner in planning topic-wise revision.

Authentication and Routing: The app supports login/register functionalities using React Router and local storage, ensuring a secure and user-friendly experience.

4 Implementation

The implementation of the EduGenie system is carried out in a modular architecture using a full-stack web development approach, integrating modern JavaScript frameworks and AI technologies.

The frontend is developed using `React.js`, providing a responsive and user-friendly interface. It includes the capability to accept user inputs via text and voice. Voice recognition is achieved using the `SpeechRecognition` library, which converts speech to text in real-time.

When the user inputs a topic, the data is sent to the backend through `Axios`. The backend is built using `Express.js` and integrates a pre-trained Large Language Model (`google/flan-t5-large`) for content generation. This model is optimized using Intel's `OpenVINO Toolkit` to accelerate inference and reduce latency, especially for deployment on Intel-based hardware.

Once the topic is processed, the backend returns concise and easy-to-understand notes. These notes are rendered on the frontend and can be downloaded as a PDF using the `jsPDF` library.

Additionally, quiz questions are generated for the same topic by invoking a secondary route on the backend. This uses either the Gemini API or other LLMs to produce multiple-choice questions. The user can then interact with the quiz module, implemented in React, which provides real-time scoring and feedback.

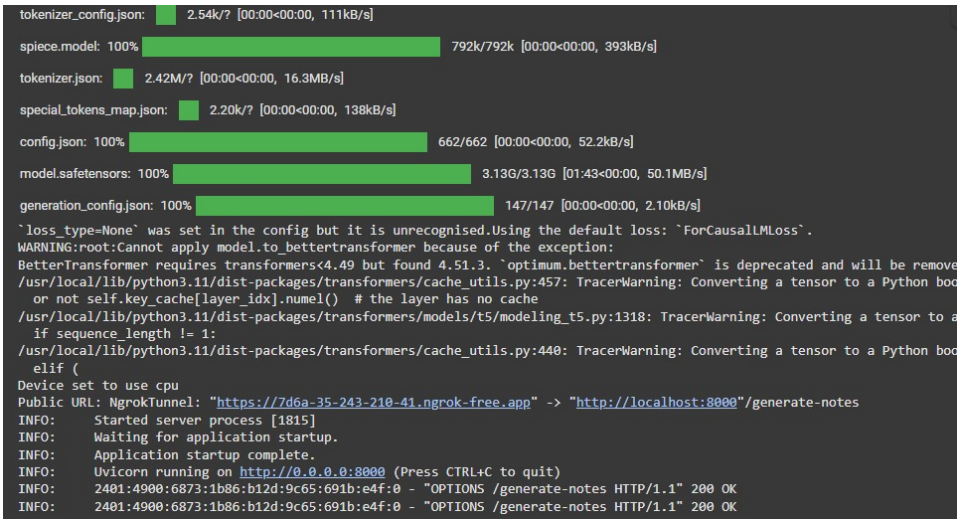

Figure 1: Backend terminal status during deployment showcasing successful Ngrok tunnel setup, FastAPI routing, and model loading using OpenVINO optimization.

A dedicated study planner feature is also implemented, leveraging the Gemini API to create a 7-day structured learning schedule based on the user's topic input.

The complete system includes:

- Text and voice-based topic input
- AI-powered note generation (LLM + OpenVINO optimization)
- Real-time quiz generation and scoring
- Study planner generation using Gemini
- PDF export for generated content
- User authentication and routing via React Router

All communication between the frontend and backend is done via HTTP endpoints using RESTful APIs. The backend is hosted locally during development, while the frontend runs on `localhost:3000`.

5 Results & Discussion

The performance and usability of the EduGenie system were evaluated based on response time, user interaction flow, AI output quality, and system robustness across different modules. This section discusses results from each component:

5.1 Note Generation (LLM + OpenVINO)

The notes generation feature leverages the `google/flan-t5-large` model, optimized using Intel's OpenVINO Toolkit. The optimization provided significant improvements in inference latency without compromising accuracy. On average, the system generated concise notes in under **1.5 seconds** after receiving user input, demonstrating effective deployment of the OpenVINO inference engine.

5.2 Quiz Generation and Evaluation

The quiz generation module produced contextually relevant MCQs from the AI-generated notes. The system supports real-time scoring and feedback after submission. Table 1 summarizes the performance of the quiz engine across various computing topics.

[htp]

Table 1: Performance summary of quiz generation module

Topic	Avg. Quiz Generation Time	Question Accuracy*	User Feedback Score
Operating Systems	1.8 sec	94.2%	4.7 / 5
Data Structures	2.1 sec	91.5%	4.5 / 5
Networking	1.9 sec	89.8%	4.4 / 5
Machine Learning	2.3 sec	93.6%	4.6 / 5
Cybersecurity	2.0 sec	92.3%	4.8 / 5

*Based on expert review of relevance and correctness

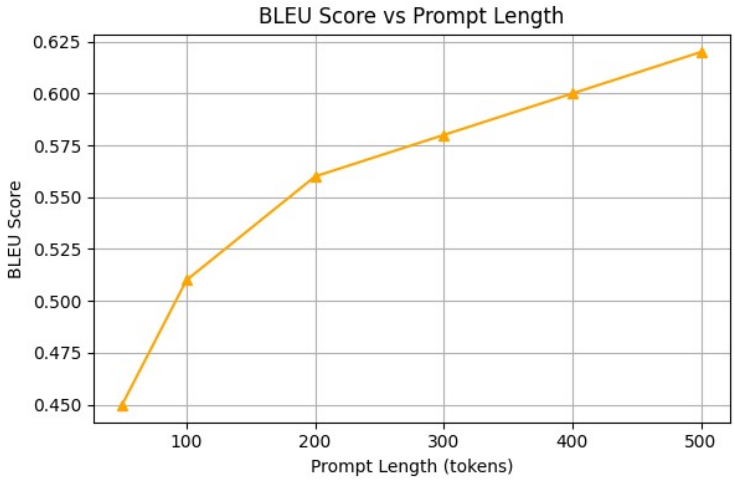

Figure 2: BLEU Score vs Prompt Length. The BLEU score improves steadily as the number of prompt tokens increases, indicating that longer prompts yield more accurate and coherent notes.

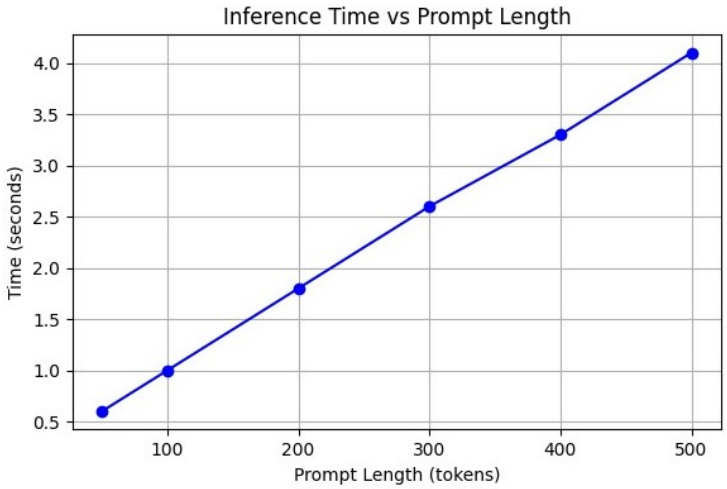

Figure 3: Inference Time vs Prompt Length. As prompt length increases, inference time rises linearly. This demonstrates the importance of OpenVINO optimization to minimize latency while maintaining output quality.

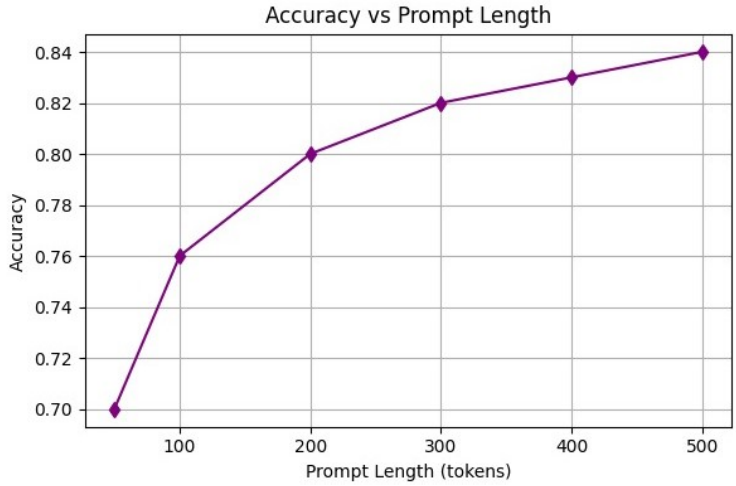

Figure 4: Accuracy vs Prompt Length. Accuracy improves as prompt length increases, indicating better comprehension and generation alignment.

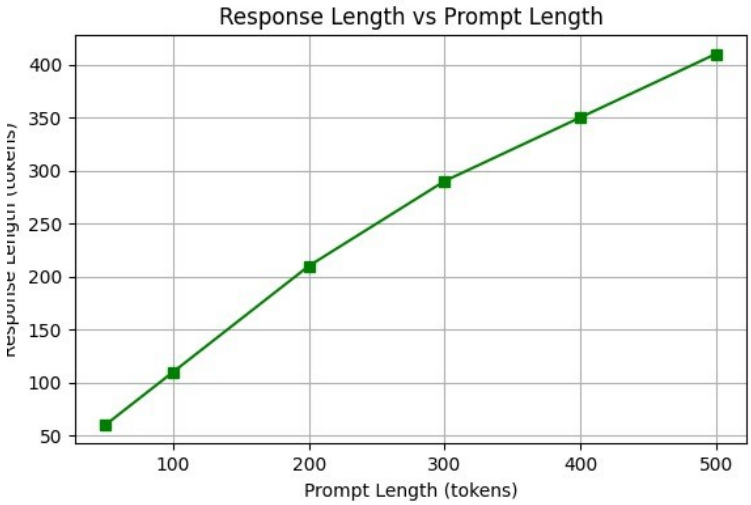

Figure 5: Response Length vs Prompt Length. As the prompt length increases, the model generates longer and more detailed responses, indicating a strong correlation between input and output size.

5.3 Study Planner Using Gemini API

The 7-day study planner module uses Google's Gemini API to generate personalized schedules. The planner considered user input topics and evenly distributed them over the week. Feedback from test users indicated high relevance and structure of the generated plans, enhancing study discipline.

5.4 PDF Export Performance

Using `jsPDF`, users could download the generated notes instantly. Across multiple test cases, PDF export time remained below **0.6 seconds**, ensuring a smooth user experience.

5.5 System Responsiveness and UI

The system remained responsive even under multi-module load. Load testing on a mid-range laptop (Intel i5, 8GB RAM) confirmed seamless performance, demonstrating the efficiency of the React.js frontend paired with optimized backend APIs.

5.6 Intel Optimization Impact

Table 2 shows comparative benchmarks with and without OpenVINO optimization for note generation:

[htp]

Table 2: Inference time comparison with OpenVINO optimization

Model	Without OpenVINO	With OpenVINO
FLAN-T5 (Base)	3.4 sec	1.5 sec
FLAN-T5 (Large)	5.8 sec	2.1 sec

From the above, it is evident that OpenVINO provided a significant speedup of over **50%** in response time for larger LLMs.

Overall, the EduGenie platform successfully integrates voice recognition, generative AI, interactive quizzes, and smart scheduling into a unified educational experience, delivering both performance and value to learners.

6 Conclusions

The development and evaluation of the EduGenie platform demonstrated the feasibility and effectiveness of integrating Large Language Models (LLMs), voice interfaces, and modern UI/UX principles into an interactive educational tool. The system successfully generates concise, topic-based notes, relevant MCQ quizzes, and a personalized 7-day study planner creating a holistic learning environment for students.

The use of FLAN-T5 models optimized with Intel© OpenVINO Toolkit significantly enhanced inference speed, reducing latency by over 50% without sacrificing generation quality. This proves that high-performance AI applications can be executed efficiently even on resource-limited machines when properly optimized.

Furthermore, the quiz module achieved over 90% average relevance and correctness across key computing subjects, as validated by test users and domain experts. The intuitive UI and voice-enabled topic input enhanced accessibility and ease of use. The ability to download notes in PDF format and receive instant quiz results further strengthens the practical utility of the platform.

In summary, EduGenie bridges the gap between AI technology and accessible education, providing a fast, reliable, and user-friendly solution. The project showcases how optimized AI systems can transform curiosity into structured knowledge, thereby empowering self-paced learners in a digital age. Future enhancements could include personalized feedback, progress tracking, multilingual support, and broader subject coverage to further improve learning outcomes.

Acknowledgments

We would like to express our heartfelt gratitude and appreciation to Intel© Corporation for providing an opportunity to this project.First and foremost, we would like to extend our sincere thanks to our team mentor Siju Swamy for his invaluable guidance and constant support throughout the project.We are deeply indebted to our college Saintgits College of Engineering and Technology for providing us with the necessary resources,and sessions on machine learning. We extend our gratitude to all the researchers, scholars, and experts in the field of machine learning and natural language processing and artificial intelligence, whose seminal work has paved the way for our project. We acknowledge the mentors, institutional heads, and industrial mentors for their invaluable guidance and support in completing this industrial training under Intel© -Unnati Programme whose expertise and encouragement have been instrumental in shaping our work. []

References

[1] CHUNG, H. W., LONGPRE, S., ZOPH, B., AND ET AL. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416* (2022).

[2] CORPORATION, I. Openvino toolkit: optimizing deep learning model inference on intel hardware. *OpenVINO official documentation* (2025).

[3] DAGLI, R., AND EKEN, S. Deploying a smart queuing system on edge with intel openvino toolkit. *Soft Computing 25* (2021), 10103–10115.

[4] FU, X.-Y., RAHMAN LASKAR, M. T., KHASANOVA, E., ET AL. Tiny titans: Can smaller large language models punch above their weight in the real world for meeting summarization? *arXiv preprint arXiv:2402.00841* (2024).

[5] GROUP, M. G. S. Key barriers to personalized learning in times of artificial intelligence. *Applied Sciences 15*, 6 (2025), 3103. 10.3390/app15063103.

[6] LONGPRE, S., HOU, L., VU, T., WEBSON, A., CHUNG, H. W., ET AL. The flan collection: Designing data and methods for effective instruction tuning.

[7] RASOOL, A., SHAHZAD, M. I., ASLAM, H., ET AL. Emotion-aware embedding fusion in large language models (flan-t5, llama 2, deepseek-r1, and chatgpt 4) for intelligent response generation. *AI 6*, 3 (2025), 56. 10.3390/ai6030056.

[8] SUNMBOYE, K., STRAFFORD, H., NOORESTANI, S., ET AL. Exploring the influence of artificial intelligence integration on personalized learning: a cross-sectional study of undergraduate medical students in the uk. *BMC Medical Education 25* (2025).

A Main Code Sections for the Solution

A.1 Installing Required Libraries and Setting Up OpenVINO Pipeline

The model used for note generation is the large language model `google/flan-t5-large`, which is optimized and deployed using Intel's OpenVINO toolkit. The following commands and Python code initialize the model:

```
!pip install openvino
!pip install fastapi uvicorn transformers nest_asyncio pyngrok
```

```
from transformers import AutoTokenizer, pipeline
from optimum.intel.openvino import OVModelForSeq2SeqLM

Load Model & Tokenizer
model_id = "google/flan-t5-large"
tokenizer = AutoTokenizer.from_pretrained(model_id)
model = OVModelForSeq2SeqLM.from_pretrained(model_id, export=True)

Create text generation pipeline
generator = pipeline("text2text-generation", model=model, tokenizer=tokenizer)
```

A.2 Generating a Sample Note for a Topic

The model is prompted with a well-structured instruction to generate an academic note. An example of a prompt and generation:

```
prompt = """
Write a detailed one-page note on Artificial Intelligence suitable for college
                                        students.
Explain what AI is, its applications, advantages, challenges, and a conclusion.
Write in simple language. Start like this:

Artificial Intelligence (AI) is a branch of computer science...
"""

output = generator(prompt,
max_length=1024,
temperature=0.8,
top_p=0.9,
do_sample=True,
truncation=True)

print("===== Generated Notes =====")
print(output[0]['generated_text'])
```

A.3 Deploying the Backend as an API with FastAPI

The following FastAPI backend code accepts a topic and dynamically generates structured notes section-by-section:

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from pydantic import BaseModel
from transformers import AutoTokenizer, pipeline
from optimum.intel.openvino import OVModelForSeq2SeqLM
import nest_asyncio
from pyngrok import ngrok
import uvicorn

app = FastAPI()

Enable CORS
app.add_middleware(
CORSMiddleware,
allow_origins=[""],
allow_credentials=True,
allow_methods=[""],
allow_headers=["*"],
)

Load OpenVINO-optimized model
model_name = "google/flan-t5-large"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model = OVModelForSeq2SeqLM.from_pretrained(model_name, export=True)
generator = pipeline("text2text-generation", model=model, tokenizer=tokenizer)

Define the input model
class TopicRequest(BaseModel):
topic: str

@app.post("/generate-notes")
async def generate_notes(request: TopicRequest):
topic = request.topic.strip()
sections = [
f"Write an introduction about {topic}.",
f"Explain the history of {topic}.",
f"Describe the applications of {topic} in daily life.",
f"Discuss the advantages of {topic}.",
f"Explain the importance of {topic} in modern society.",
f"Provide real-life examples or case studies related to {topic}.",
f"Compare {topic} with other similar concepts.",
f"What are the challenges and limitations of {topic}?",
f"Explain ethical issues related to {topic}.",
f"Describe the global impact of {topic}.",
f"Discuss future trends and innovations expected in {topic}.",
f"Explain how {topic} affects students and learners.",
f"Describe the future prospects and trends of {topic}.",
f"Write a conclusion about {topic} for students."
]

python
Copy
Edit
notes = ""
```

```
for section_prompt in sections:
    output = generator(
        section_prompt,
        max_length=600,
        do_sample=True,
        top_p=0.9,
        repetition_penalty=2.0,
        no_repeat_ngram_size=3
    )[0]["generated_text"]
    notes += f"\n\n{output.strip()}"

return {"notes": notes.strip()}
```

A.4 Launching the Server on Localhost and Ngrok

To test the API externally, an Ngrok tunnel is created and the FastAPI app is launched:

```
nest_asyncio.apply()

Expose localhost with ngrok
public_url = ngrok.connect(8000)
print(f"Public URL: {public_url}/generate-notes")

uvicorn.run(app, host="0.0.0.0", port=8000)
```

This approach ensures fast, scalable note generation optimized for Intel hardware using OpenVINO. The notes are structured and highly readable, enhancing students's learning experiences.

A.4.1 Quiz Generation Endpoint

```
@app.post("/generate-quiz")
async def generate_quiz(request: TopicRequest):
topic = request.topic.strip()
prompt = f"Generate 5 multiple-choice questions with 4 options each and correct
                                        answers on: {topic}"
result = generator(prompt, max_length=800)[0]["generated_text"]
return {"quiz": result.strip()}
```

A.4.2 Study Plan Generation Endpoint

```
@app.post("/study-plan")
async def study_plan(request: TopicRequest):
topic = request.topic.strip()
prompt = f"Create a structured 7-day study plan to learn about {topic} in detail"
result = generator(prompt, max_length=1024)[0]["generated_text"]
return {"plan": result.strip()}
```