A Paradigm Shift in Educational Content Creation: Leveraging Generative AI for On-Demand Course Generation

Sahil Pulikal[1], Dr. Nita Patil[2], Dr. C. M. Raut[3], Dr. Sanjay M. Patil[4]

[1,3,4]Department of Computer Engineering, Datta Meghe College of Engineering, Airoli, Navi Mumbai, India
[2]Department of Computer Engineering, K. C. College of Engineering and Management and Research Studies, Thane, India.

Abstract
Applications for creating courses have the power to completely change how educational content is created. But the existing systems have a number of drawbacks, including limited flexibility such as not being able to add unique important topics, a limited range of topics, no integration of YouTube videos, and no interactive tests. In order to overcome these drawbacks, a novel system is presented in this work. With the help of our system, users may create customized courses covering a variety of topics and include interactive tests and YouTube video incorporation.
The suggested system makes it possible to create highly personalized courses in a variety of academic subjects. It evaluates user input to build thorough course outlines that cover important topics and subtopics. It does this by utilizing the most recent robust Generative AI models, such as OpenAI's GPT 4o mini, and smart prompt engineering approaches. In addition, the software offers interactive tests and smoothly incorporates relevant YouTube videos to improve student comprehension and engagement.
The architecture as well as functionality of the suggested system are covered in detail in this paper. This system has the ability to significantly change the educational landscape by providing teachers and students with a more tailored and interactive learning environment.

Keywords:
Course generator, Generative AI, Artificial Intelligence, Education

1. INTRODUCTION

One of the most innovative and revolutionary areas of AI is generative AI, which enables computers to produce material on their own while imitating human creativity and problem-solving abilities [21, 22]. This paper is an important first step toward reimagining the creation, organization, and improvement of instructional content. It is a feature-rich full stack application that creates course structures dynamically using Generative AI, loads them with pertinent YouTube videos, and improves learning utilizing concept-check questions.

The process of developing educational courses has traditionally been laborious and taking time. Teachers often find it challenging to put together a course structure that is coherent and successfully conveys material. Moreover, maintaining the relevance and interest of course material is a never-ending challenge. Accessing pertinent outside resources, like YouTube videos, is essential as the internet develops as a repository for educational content. Ultimately, quizzes—a preliminary evaluation tool—are essential for gauging student understanding and consolidating prior knowledge. The purpose of this paper is to address these intricate learning challenges.

This paper's primary goal is to lay the groundwork for a state-of-the-art, user-friendly platform that makes it simpler to develop comprehensive courses utilizing generative AI. It can achieve this by making use of the following essential components:

1. Artificial Intelligence-Driven Course Outline Generation: Using state-of-the-art AI models like Open AI's GPT 4o mini, the computer analyzes the input keywords to generate comprehensive course structures. This makes it easier to create courses and allows for a logical and effective arrangement for the content.
2. Integration of YouTube Videos: The platform dynamically retrieves and embeds pertinent YouTube videos into the course framework. This enhances learning by providing students with access to multimedia resources that supplement written content and take into account different learning styles.
3. Concept Testing Quiz Questions Generator: To boost comprehension and engagement, the software automatically generates concept-check quizzes. Teachers can assess how well students are doing and emphasize key concepts because these examinations correspond with the course material.

The technique for this paper analyzes user input, including input subjects and subtopics, using OpenAI's GPT 4o micro model to create comprehensive course outlines. Users provide the course title and subtopics, and the OpenAI GPT 4o mini model is guided in generating units, chapters, and questions by means of advanced prompts. To improve learning, the paper incorporates YouTube video retrieval and generates summaries based on video transcripts. Secure access and limitless course creation are made possible by user login and payment connection. Navigating and interacting with course content is made easier with an intuitive interface.

This paper is extremely pertinent to the discipline of education:

1. Efficiency: Teachers may concentrate on instructional excellence and creativity while reducing their time and energy by automating the course creation process.
2. Quality: Adding outside multimedia resources to a course, such YouTube videos, improves the material and creates more engaging and applicable real-world scenarios.
3. Assessment: Through the provision of immediate feedback and chances for reinforcement, formative assessment, like concept-check quizzes, improves student learning outcomes.

In many respects, this application advances technology for education. To begin with, it goes much beyond what is possible with existing programs by dynamically creating vast course structures with a diverse range of subjects using sophisticated artificial intelligence models, such OpenAI's GPT. Furthermore, it automatically finds relevant YouTube videos, which improves learning and accommodates different learning styles. Most notably, the application offers an interactive assessment method not seen in other solutions by using an inventive process to create quizzes straight from transcripts of YouTube videos. This strengthens the app's potential to revolutionize education by enabling effective and high-quality

content creation. Additionally, the application creates interactive questions for tests automatically utilizing video transcripts, a unique feature not seen in other applications that raises the bar. This feature, which incorporates evaluation questions into educational resources smoothly, promotes greater participation and information retention.

Ultimately, this application represents a major breakthrough in educational technology since it enhances the process of learning and instruction through the use of AI-driven course creation, YouTube incorporation, and quiz development. The process of designing, developing, and testing this revolutionary platform will be covered in depth in the upcoming chapters of this article. The ultimate goal is for it to enhance education globally by increasing its effectiveness, accessibility, and appeal to learners.

The goal of this paper is to give readers a thorough grasp of the AI Course Generator architecture. Chapter 2 discusses the relevant works. A few of the current systems' inadequacies are highlighted in Chapter 3. In Chapter 4, the principles and foundation of generative AI are covered. Chapter 5 goes into detail about the methodology. Chapter 6 discusses the results and discussion. Chapter 7 discusses the conclusion and future scope.

2. LITERATURE REVIEW

An extensive overview of generative artificial intelligence and its many uses is provided in this section.

R. AlAli and Y. Wardat [1] analyzes the potential of Generative Artificial Intelligence (AI) to revolutionize education. The authors acknowledge the benefits of AI, such as individualized learning and enhanced student engagement, but warn that incorporating AI into educational settings is not without its obstacles. These problems include negotiating ethical quandaries, protecting data privacy, reducing algorithmic biases, and adapting to educators' changing roles. The paper proposes a collaborative strategy that includes educators, legislators, and technologists to guarantee AI integration is responsible and beneficial to all students.

R. Kaplan-Rakowski [2] examines educators' viewpoints on generative AI (GAI) in the classroom, with a specific emphasis on ChatGPT. The writers chart the development of AI in education, stressing the benefits and drawbacks of implementing GAI. A survey of 147 teachers with varying backgrounds was used in the study to find out how they felt about integrating GAI, where they were in the adoption process, and how often they used GAI in their instruction. According to the data, most teachers have good opinions about GAI's potential in the classroom and think it's a useful tool for both students' learning and their own professional growth. Additionally, the study shows a relationship between the frequency of usage of GAI and the degree of integration among teachers. The authors address the significance of these findings in their conclusion, highlighting the necessity of professional development and teacher training to facilitate the ethical and successful incorporation of GAI into teaching practices.

A. Ghimire et al. [3] explore the experiences and attitudes of university instructors concerning the utilization of Generative AI-based tools in education, such as ChatGPT. The study examines the factors that influence these attitudes, as well as educators' perceptions of the possible impact of AI on teaching and learning. The authors utilized a mixed-methods approach, conducting surveys and interviews with university faculty members from several departments. The findings show that educators are usually favorable about these tools, citing benefits such as increased efficiency and individualized instruction. However, concerns about academic integrity, potential cheating, and the stifling of creativity persist. The paper finishes by underlining the importance of adapting to AI in education while carefully assessing its consequences and minimizing potential hazards.

M. Alier et al. [4] highlight the rapid progress and use of Large Language Models (LLMs), such as ChatGPT, which represents a significant shift in AI's impact on education. The authors believe that this "GenAI moment" is shifting from a deceptive to a disruptive phase, emphasizing both the potential and issues that LLMs provide in education, such as tailored learning experiences and academic integrity concerns. It dives more into the technical features of LLMs, their emerging capabilities, and the role of open-source models in pushing innovation in the sector, particularly in education. It concludes by exploring many elements of GenAI's integration into education, such as its ethical implications, evaluation options, and the necessity for secure and trustworthy AI applications in learning environments.

Thomas [5] investigates how generative artificial intelligence (GenAI) techniques such as ChatGPT may transform higher education. The author conducts focus group interviews with university students to explore student perceptions on how GenAI affects learning outcomes, methods of teaching (pedagogies), and evaluation. The findings suggest that future higher education should highlight new learning goals such as AI literacy and adaptability abilities, stress multidisciplinary and maker-centered learning techniques, and shift toward assessment focusing on in-class activities and real-world problem-solving utilizing GenAI. The paper emphasizes the importance of rethinking higher education methods and regulations in order to better prepare students for a GenAI-shaped future workforce.

R. AlAli et al. [6] examines the potential for generative artificial intelligence (AI) to transform education. They present a thorough review of the benefits, limitations, and ethical implications of employing generative AI in educational settings. Key issues include individualized learning, task automation, ethical considerations about bias and data privacy, and the value of professional development for educators. Finally, the study intends to give educators and policy makers practical advice as well as a theoretical framework for successfully and responsibly integrating generative AI into education to improve teaching and learning experiences.

L. Bonde [7] examines how Generative AI (GenAI) can transform education. The author contends that, despite GenAI's acknowledged promise, its practical application in educational institutions remains limited. It emphasizes individualized learning, feedback, interactivity, assessment, and professional growth as essential elements of a GenAI system for education. It also discusses the obstacles that come with GenAI adoption, such as ethical concerns, privacy issues, and the necessity for strong policies. Finally, it presents a conceptual design for a comprehensive GenAI system, with the goal of bridging the gap between individual GenAI experiments and widespread institutional adoption in higher education.

D. Griffiths [8] examines the impact of large language models (LLMs), particularly generative AI, on education. The authors claim that the fast rise of LLMs calls into question existing educational paradigms, particularly the "transmission" model, which sees learning as the transfer of knowledge. They propose that LLMs demonstrate the weaknesses in this model by showing that, even passing exams, AI lacks actual knowledge. Instead, they advocate for a "coordination" paradigm, which views learning as a collaborative process of knowledge construction. They argue that adopting this paradigm, as well as using LLMs as instruments for inquiry and

support, provides a more successful approach to teaching in the age of AI. Finally, the authors advocate for a rethinking of educational processes, asking institutions to adapt to the changing landscape made possible by AI.

A. Alammari [9] investigates the integration of generative AI (GAI), specifically tools such as ChatGPT, into Saudi higher education systems. The research used a mixed-methods approach, combining quantitative data from surveys with qualitative insights from educator interviews. Key findings show a favorable relationship between instructors' understanding of GAI and its deployment in the classroom, demonstrating a willingness to utilize this technology. While educators recognize GAI's potential benefits for personalized learning and professional growth, they also raise concerns about plagiarism, over-reliance on GAI, and job displacement. Overall, the paper emphasizes the necessity for a balanced approach to GAI integration, noting both its potential and the obstacles that must be carefully considered.

N. McDonald et al. [10] analyzes the impact of generative artificial intelligence (GenAI), namely ChatGPT, in higher education. The paper examines GenAI policies at 116 major US universities to better understand how these institutions lead professors in this fast growing technology. According to the data, the majority of universities promote the use of GenAI in the classroom by giving extensive guidance, sample syllabi, and even curriculum. This acceptance, however, is frequently accompanied by worries about the ethical implications, student privacy, and the need for educational changes to accommodate GenAI. Notably, the study found that, while instruction focuses on writing and research, there is no emphasis on GenAI's impact on STEM areas. It concludes by emphasizing the need for additional research and critical assessment of GenAI's role in education, advocating for responsible implementation based on ethical considerations and fairness.

A. Ghimire [11] investigates how artificial intelligence (AI) can be used to summarize legal papers, making legal knowledge more accessible. It also looks into instructors' perspectives of AI tools, finding overall positive but emphasizing the importance of adequate training and resources. It also examines the policy landscape surrounding AI in education, highlighting the need for more comprehensive norms. Furthermore, the study investigates the usage of AI tools such as ChatGPT in programming courses, revealing their beneficial impact on student learning. Finally, it explores how educators' attitudes toward these tools are influenced by their perceived utility and ease of use.

D. Grover [12] examines how Generative Artificial Intelligence (GenAI), with its potential to generate fresh content, can transform education. It goes into GenAI's ability to customize learning, automate tasks, and improve accessibility, using theoretical foundations and real-world examples. However, it acknowledges issues such as data privacy, ethical AI use, and equity. The paper concludes by discussing potential options for GenAI in education, such as advances in adaptive learning, integration with AR/VR, and ethical implications for responsible AI use.

H. Li et al. [13] explores the possible impact of integrating generative AI (GenAI), such as big language models, into adaptive learning systems. The authors suggest that GenAI's ability to generate diverse and dynamic content such as tailored questions, learning materials, and even simulated learning experiences can considerably improve the efficacy of adaptive learning systems. It gives a thorough assessment of existing adaptive learning strategies, focusing on how GenAI might be used to improve tasks like learner profile creation and material recommendation. However, the authors highlight problems such as the possibility of AI "hallucination" and the necessity to address biases in AI-generated content. Finally, this paper seeks to promote the creation of GenAI-powered adaptive learning systems that are successful, equitable, and prioritize human assistance in the educational process.

M. Resnick [14] explores both the risks and opportunities associated with applying generative AI into learning environments. While AI has the ability to personalize learning and foster creativity, the author contends that many present applications stress rote learning and "close-ended problems" above student agency and meaningful cooperation. He calls for a "constructionist" approach, in which AI technologies are utilized to assist project-based learning that is motivated by student passion and supported by peers in a playful setting, ensuring that AI enhances, rather than reduces, human connection. He demonstrates these arguments by looking at the impact of generative AI on coding instruction, emphasizing the possible benefits and drawbacks of conversational coding. Finally, he emphasizes that the path forward is dependent on making deliberate decisions that correspond with a human-centered vision of education.

G. Kurtz et al. [15] looks into the significant implications of generative AI (GenAI), such as ChatGPT, in higher education. The authors believe that while new technology has the potential to transform teaching and learning, it also raises ethical considerations and necessitates revised teaching approaches. It gives a comprehensive study of GenAI's potential benefits, such as individualized learning and increased student engagement, while also admitting dangers such as AI bias and potential cheating. Importantly, the authors present a four-stage model for ethical GenAI adoption by faculty, highlighting the importance of training and a systematic approach to incorporating these powerful tools into educational contexts. Finally, the paper emphasizes the importance of higher education institutions proactively adapting to GenAI's quickly expanding ecosystem in order to remain relevant and prepare students for the future.

D. E. Salinas-Navarro et al. [16] investigates how generative artificial intelligence (GenAI) tools, such as ChatGPT, can be strategically integrated into higher education to improve teaching and learning. The authors contend that authentic assessment, which stresses real-world application of information, and experiential learning, which promotes active knowledge production, provide useful foundations for this integration. It looks at how GenAI can be used to develop desired learning goals, design compelling teaching and learning activities, and create effective assessment tasks, all while adhering to constructive alignment principles to ensure pedagogical coherence. It concludes that GenAI tools can be important assets in creating a more dynamic and effective learning environment if used appropriately and ethically.

I. Pesovski et al. [17] explores the ability of large language models (LLMs), specifically OpenAI's GPT-4, to provide individualized learning experiences in a software engineering course. They explain how they incorporated an AI-powered content generation tool into an existing learning management system (LMS) to generate course materials in three different styles: traditional computer science teacher, Batman, and Wednesday Addams. The study, which included 20 students, used questionnaires to measure student participation with various content versions as well as their perceptions. Preliminary findings indicate that, while students preferred traditional content styles, the availability of different, AI-generated resources boosted overall study time and was especially beneficial for students who struggled with the material.

S. Ivanov et al. [18] explores how generative AI tools' perceived strengths, shortcomings, dangers, and rewards

influence their adoption by higher education students and instructors. The Theory of Planned Behavior (TPB) is used in the study to explain how these views impact attitudes, subjective norms, and perceived behavioral control toward GenAI use, eventually influencing intention and actual usage. The authors examined 130 lecturers and 168 students from diverse nations, and discovered that perceived strengths and benefits were the most important predictors of positive attitudes and plans to adopt GenAI. Surprisingly, the study found that perceived limitations and hazards had a smaller impact on adoption than expected. Based on their findings, the authors propose various management and policy implications for higher education institutions seeking to encourage the responsible and successful integration of GenAI in teaching and research.

L.-H. Wong et al. [19] identifies three major themes discussed throughout the issue. Theme 1 investigates the growing link between humans and AI, focusing on the possibility for collaborative learning environments. Theme 2 dives into the technological and pedagogical opportunities and problems given by AI, highlighting the necessity of critical analysis and user perception. Finally, Theme 3 focuses on teacher perspectives on AI adoption, emphasizing the importance of supporting instructors as they navigate these new technologies. Finally, this study advocates for a balanced and thoughtful strategy to incorporate AI into education, as well as additional research in this fast changing field.

R. Sajja et al. [20] explores the changing role of hackathons in the age of AI, with a specific emphasis on the incorporation of generative AI technologies such as ChatGPT. The paper investigates how new technologies affect typical hackathon dynamics, students' technology choices, and project development. It emphasizes the opportunities provided by AI, such as improved learning experiences and higher project sophistication, while also noting the challenges, such as ethical concerns and potential over-reliance on AI tools. Through a case study of the HACKUIOWA 2023 event, the source examines survey data to better understand student perspectives and usage patterns of AI in hackathons, providing insights into the perceived value and possible influence of these tools on coding techniques and collaboration. Finally, the article intends to provide a road map for incorporating AI into future hackathons, highlighting the importance of taking a balanced approach that capitalizes on AI's benefits while maintaining the fundamental principles of creativity, collaboration, and skill development.

3. SHORTCOMINGS OF CURRENT SYSTEMS

Below is a list of several shortcomings in the current systems that have been found:

Inadequate customization: It's common for current course generator applications to be unable to produce extremely customized courses. This is because their training usually consists of extensive databases of previously created courses, which might not accurately reflect the unique requirements of every user.

Subject coverage is constrained: The majority of course generation software only includes topics in science, math, and English. This suggests that anyone looking to develop courses on more specialized subjects would find them inappropriate.

Absence of YouTube Video Incorporation: This would improve learners' comprehension of the material and is a shortcoming of the present platforms.

Absence of interactive quizzes: The majority of course creation programs do not produce lively or interesting quizzes. Children may find it difficult to appropriately learn and remember information as a result.

Complex UI: makes it difficult for users to design courses.

In order to overcome these limitations, our paper added several new features, such as:

Options for customization: By deciding on the exact subjects they wish to learn, students can make their courses uniquely their own.

broad range of topics: Users will have the ability to design courses that address a broad range of issues.

Incorporation of YouTube Videos: Students will have a deeper understanding of the subject by watching videos on YouTube.

Multiple-choice interactive quizzes: These kinds of interactive tests help learners recall and assimilate information.

Simple user interface: Anyone may design quizzes with ease thanks to a simple user interface.

4. FUNDAMENTALS OF GENERATIVE AI

Generative AI" is the branch of artificial intelligence that focuses on developing algorithms that can generate new types of information, including text, images, audio, and video. Generative AI models are trained on vast amounts of pre-existing content, and they eventually learn to generate new content that is similar to the training set. Generative AI models are capable of a wide range of tasks [21], including:

4.1 Text generation models

Text generation models are capable of producing a variety of text formats, such as code, creative writing, and news articles. Their foundations are frequently large language models (LLMs) trained on enormous text and code datasets. LLMs are capable of producing original and creative text formats in addition to material that is comparable to the text they were taught.

Here are a few examples of text-generating models:

GPT-3: OpenAI developed the GPT-3 LLM. One of the most powerful text generation models available, it can produce text in a wide range of formats, such as emails, letters, code, poetry, and musical compositions. [21] [23]

Codex: Codex is an LLM created by Google AI. Its primary purpose is to generate code in multiple programming languages, such as Python, JavaScript, Go, and C++. [21]

Google AI has developed a machine learning model called LaMDA. It provides customers with thorough and insightful answers to questions, regardless of how strange, complicated, or open-ended they may be. [21]

4.2 Image generation models

Models for image creation can produce realistic images, including drawings, cartoons, and photos. They are usually created with deep learning methods like generative adversarial networks (GANs). Two neural networks that have been taught to compete with one another are called GANs. The initial neural network produces fresh visuals. The 2nd neural network, the discriminator, attempts to discern between the real and fraudulent images produced by the generator. The generator becomes more adept at producing pictures that resemble real photographs over time.

Here are a few instances of models for creating images:

Imagen: The model used by Google AI to create photos is called Imagen. It has the ability to translate written descriptions into lifelike images. [21]
DALL-E 2: The DALL-E 2 image-generating model was created by OpenAI. It has the ability to alter existing photos and produce lifelike images from textual descriptions. [21]
Parti: Parti is a photo-editing model created by Google AI. It can be used to make duplicate copies of an image and replace missing pixels in pictures. [21]

4.3 Video generation models

Movies, music videos, and photorealistic videos are just a few of the types of videos that can be produced by video generating models. Even though they are still in the early phases of improvement, these advancements have the power to completely transform the way that videos are made.

Here are a few examples of models that generate videos:
DALL-E 2: It allows for written explanations to produce videos. [21]
Imagen: Imagen can create movies using written information, however the output is not as realistic as what DALL-E 2 can create. [21]

4.4 Text-to-3D models

By transforming words, text-to-3D models can produce creative and realistic 3D visuals. Dreamfusion and Magic3D are two examples [21].

4.5 Speech synthesis

GAI voice synthesizing models are capable of producing speech that sounds human. Usually, deep learning methods like RNNs, or recurrent neural networks, are used to build these models. For example, text can be converted into speech that sounds human using the Google Cloud's Text-to-Speech API [21].

4.6 Overview of generative AI backbones

The underlying structures that serve as the foundation for generative AI models are referred to as generative AI backbones. They may produce a range of content kinds, such as text, graphics, and audio, and they offer a fundamental structure to teach generative models.

Each generative AI backbone has certain advantages and disadvantages. Though they are unstable and challenging to train, GANs are capable of producing written content, audio recordings, and images of excellent quality. Compared to GANs, VAEs generate written, audio, and picture information of poorer quality, but they are easier to train and more trustworthy. Transformers are capable of producing written content, visual content, and code of excellent quality, but their computational training costs are higher than those of GANs or VAEs.

The kind of generative AI foundation that is employed depends on the specific task at hand. If producing written material, audio recordings, or pictures of excellent quality is still the main objective, then GANs might be the ideal method. If you have the computational capacity and want to generate high-quality text, images, or code, transformers might be the best option. If producing high-quality text, pictures, or audio with minimal computational resources is the aim, or if maintaining model consistency is crucial, VAEs might be the best option.

One efficient way to train generative AI models is via generative AI backbones. They make it possible to train generative models that can generate excellent code, images, music, and literature. But it's crucial to choose the right backbone for the job at hand.

4.7 Generative Adversarial Networks (GANs)

One of the most widely used generative AI backbones is the GAN. The discriminator and generator neural networks make up a GAN. While the generator creates fresh data samples, the discriminator is in charge of differentiating between real and false data samples that it generates.

Due to the adversarial nature of GAN training, there is a constant attempt by the discriminator and generator to trick one another. While the generator aims to produce false data samples that resemble real data samples, the discriminator looks for differences between real and fake data samples.

GANs have demonstrated their ability to produce written material, audio recordings and images of excellent quality. They could still be erratic and challenging to train.

GANs are used to accurately represent people, places, and objects in deep fakes and authentic pictures for video games and movies.

4.8 Variational Autoencoders (VAEs)

VaEs are yet another well-liked generative AI foundation. One kind of VAE is an autoencoder, which is a neural network taught to duplicate its input data. During training, VAEs not only learn how to reassemble the input data, but they also pick up the latent form of the data.

A low-dimensional form of data that highlights its most crucial characteristics is called a latent representation. By collecting samples from the data's latent representation, VAEs are able to produce fresh data samples.

When it comes to training time and stability, VAEs perform better than GANs. They might still not be able to produce images, audio, and text with the same level of quality as GANs. VAEs are used in the development of new materials, customized experiences of learning for students, and the synthesis of novel pharmaceutical molecules. [21]

4.9 Transformers

Transformers are a relatively new generative AI backbone with encouraging results. Transformers are based on a particular kind of neural network that is frequently employed in machine translation: the encoder-decoder architecture.

It has been demonstrated that transformers are efficient in producing written words, visuals, and code of the highest caliber. However, they might require more processing power to train than GANs or VAEs.

When composing news articles and creating new software applications, transformers are utilized to produce realistic text and code. [21]

4.10 Benefits of Generative AI

Generative AI has several potential benefits, such as:
Increased productivity: Many tasks that are now performed by humans, such as developing marketing collateral, producing reports, and coming up with new goods and services, can be automated by generative AI. Human workers may be able to concentrate on more creative and strategic work as a result. [21, 23]
Increased creativity: By generating fresh concepts and motivation, generative AI can increase human creativity. For example, generative AI may be utilized to create original songs, tales, and commercial ideas. [21]

Individualized experiences: Thanks to generative AI, consumers can get personalized experiences. For instance, users can receive tailored learning experiences or recommendations for goods and services via generative AI. [21]
Better decision-making: Generative AI provides insights into complex data to assist people in making more informed decisions. Data patterns and future event forecasting are two applications of generative AI. [21]
Novel goods and services: It is now possible to develop previously unthinkable novel goods and services. For instance, generative AI may be utilized to create novel medications, therapeutic interventions, and educational materials. [21]

4.11 Challenges of Generative AI

Among the difficulties facing generative AI are:
Data requirements: To train, generative AI models need a lot of high-quality data. It could be costly and time-consuming to gather, classify, and handle this data.
Computational requirements: Generative AI model deployment and training may become computationally costly. Starting a small business or group could become more difficult as a result.
Biases: Generative AI models have the potential to be prejudiced since they represent the biases present in the training data. This could result in the production of offensive or dangerous content.
Misinformation: Generative AI models can be used to produce deep fakes along with various types of misinformation. This might be utilized to influence others or harm someone's reputation.
Safety: Content that is harmful or dangerous, like weapons or artificial drugs, can be created by generative AI models. Ensuring the safe and responsible application of generative artificial intelligence (AI) models is crucial.

In addition to these difficulties, ethical questions are brought up by the creation and application of generative AI. Think about how generative AI will affect society and how content will be created and shared, for instance.

4.12 Addressing the challenges of Generative AI

The issues with generative AI can be solved in a number of ways [23][24]. As an illustration:
Requirements for data: Scientists are developing cutting-edge methods for teaching generative AI models with less data. Moreover, there's a growing movement to create publicly accessible data sources that can be utilized for generative AI model training.
Computational requirements: To increase the effectiveness of generative AI models, researchers are developing new techniques and technologies. Additionally, cloud computing companies are providing services that lower the cost and simplify the process of training and implementing generative AI models.
Bias: To lessen biases in generative AI models, researchers are looking into cutting-edge techniques. In addition, it is imperative to exercise caution while handling the data that generative AI systems are trained on.
False information: Scientists are developing novel methods to identify and stop the production of deep fakes and other types of disinformation. It is also essential for informing the general population about the potential for deep fakes and other forms of false information.
Safety: To guarantee that generative artificial intelligence models are used sensibly and safely, researchers are creating novel strategies. It's still crucial to establish moral standards for the creation and application of generative AI.

5. METHODOLOGY

The following describes the technology implementation and application flow that comprise the methodology.

5.1 Application Flow details:

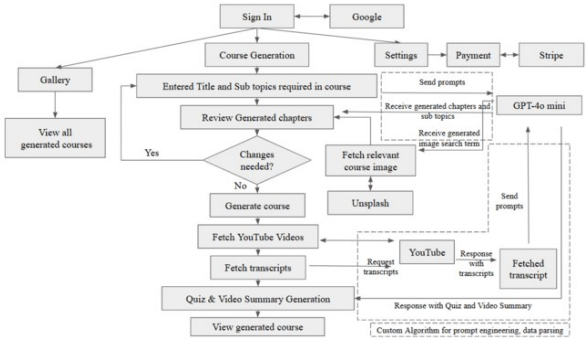

Fig 5.1. Application flow diagram

There are several components in the application flow diagram, which are shown in Fig. 5.1 and will be outlined in the following manner.

5.1.1 User Authentication:

The credentials of a user are verified and authenticated when they sign in using Google authentication. A distinct user session is created and connected to their account once authorized.

5.1.2 Course Generation:

The user enters the title of the course and any required subtopics.

To ensure that all pertinent fields are correctly filled out, user input is reviewed.

Based on user input, advanced prompts are generated and sent to OpenAI's GPT API.

The relevant chapters and subtopics are generated by the OpenAI GPT API after it has interpreted the inputs.

The user can then evaluate the chapters and subtopics that the OpenAI GPT API has generated.

The created material can be reviewed by the user, who can also modify the title and subtopics as needed.

The user is returned to the input phase if any modifications need to be made.

If the user accepts the generated content, the course creation process proceeds.

5.1.3 User Interaction with the AI Model:

Users interact with the AI model indirectly by providing information on the course creation page, which requests input such as main subjects and subtopics of the required course, prompting the system to generate subsequent prompts based on the inputs.

Sophisticated prompts that respond to user input are constructed and sent to the OpenAI GPT API.

A course structure with chapters and subtopics is constructed by the AI model based on the input query, after which the user can view the generated material.

Produced chapters and subtopics obtained from the OpenAI GPT API are displayed to the user for evaluation.

The user is permitted to revise the title and subtopics as necessary after viewing the produced material.

The received input may be utilized to enhance the produced content or generate entirely new material.

5.1.4 User Feedback:

There are two methods for integrating input from users into the program:

If the user accepts the produced material, the course can continue to be created.

In case modifications are required, the user goes back to the original phase.

5.1.5 Fetching Videos and Transcripts from YouTube:

YouTube video transcripts are retrieved asynchronously for each obtained YouTube video, enabling efficient processing. Concurrently, the YouTube Data API is utilized to recursively obtain the relevant YouTube video URLs corresponding to each chapter and subtopic, facilitating comprehensive content aggregation.

5.1.6 Quiz and Video Summary Generation:

Obtained video transcripts are utilized to construct intelligent prompts in the aforementioned steps.

These sophisticated prompts are then employed in the OpenAI GPT API to generate quizzes and YouTube video descriptions.

The quizzes and video summaries received from the OpenAI GPT API are subsequently assigned to the appropriately created chapters and subtopics.

5.1.7 Visualization of course:

The prepared course, encompassing quizzes, relevant YouTube videos, chapter and subtopic summaries, and additional materials, is presented to the user.

An intuitive user interface is provided, enabling students to interact with quizzes, video summaries, and course content in an engaging manner.

5.1.8 Payment Gateway Integration:

The system incorporates a subscription-based model enabled by the Stripe payment gateway to facilitate unlimited course creation.

Secure online transactions are processed through Stripe, granting users unrestricted access to the platform's course generation features upon successful payment, thereby ensuring seamless and secure user experience.

5.1.9 Course Gallery:

To provide a comprehensive overview of generated courses, a user-accessible gallery is implemented. Each course is visually represented by a thumbnail image dynamically fetched from the Unsplash API. The image selection process is guided by the course title, ensuring visual relevance. Essential course metadata, including title and subtopics, is displayed alongside the thumbnail, offering users a clear and concise representation of each course.

Here are some ways that the proposed approach differs from current methodologies:

1. It uses the latest AI models, such as the OpenAI GPT-4o mini model, a state-of-the-art model for developing high-quality course structures and content.
2. Contributions from users on the material to be created is used to improve the finished result.
3. It provides users with access to relevant videos on YouTube, making learning more engaging and dynamic.
4. With the use of transcripts from videos, users may create interactive assessments that help students assess their understanding of the material and reinforce key concepts.
5. It is extendable and may be used to develop courses on a range of topics.
6. It is made to be user-friendly and accessible to anyone with different levels of technical proficiency.

Limitations and Constraints of the Proposed Framework

The efficacy of the proposed course generation framework is contingent upon several factors and subject to certain limitations.

Dependency on Input Quality: Both human-provided prompts and system-generated prompts significantly influence the quality of the generated course content.

Scope Limitations: The AI model's knowledge base may not encompass every conceivable topic, potentially restricting the framework's applicability.

Potential for Bias: The generated content may be susceptible to biases present in the training data of the AI model.

Infrastructure Dependency: A stable internet connection is essential for the framework's operation, as it relies on external APIs and cloud-based services.

5.2 Specifics of technical implementation:

1. Configuring and creating a new folder for NextJS.

A new folder is configured and created for NextJS, necessitating the installation of required dependencies and initiation of a fresh NextJS project.

1.1. Set up NextJS.

NextJS is set up by creating a next.config.js file in the root, which will contain the NextJS configuration, including TailwindCSS and routing configurations.

1.2. Establish a database.

A database is established to store user and course data, with options including PostgreSQL, MongoDB, and MySQL; Aiven is recommended for creating a free cloud-based MySQL database.

1.3. Make an user interface for users.

A user interface is developed to enable user interaction with the application, allowing users to create courses, view generated courses, and upgrade to paid user status through payment processing.

1.4. For styling, Incorporating TailwindCSS.

TailwindCSS is incorporated for styling purposes, requiring the installation of TailwindCSS and subsequent styling of the application.

The installation of the Node.js SDK is necessary to utilize Tailwind CSS.

A tailwind.config.js file is prepared in the root directory following SDK installation, containing the TailwindCSS configuration.

With the configuration file in place, styling can be performed using TailwindCSS, enabling the application's visual design.

2. Incorporating the GPT-4o mini to create course materials.

The GPT-4o mini is incorporated to generate course materials, necessitating the acquisition of an API key and setup of an OpenAI account.

An OpenAI account is created and an API key is obtained by signing up for a free account at https://platform.openai.com.

The OpenAI Node.js SDK is installed using the acquired API key, followed by the initialization of the new OpenAI client.

The client is utilized to create course content, allowing for prompt engineering techniques to tailor the output according to specific requirements.

The generated course material is then created based on the outline, leveraging the OpenAI model's capabilities.

Algorithm for strict_output function (gpt.ts file) for handling and parsing OpenAI API:

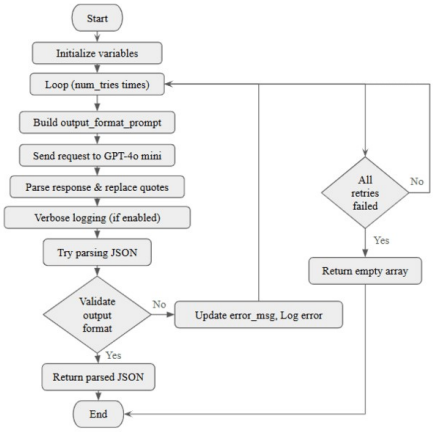

Fig 5.2.1. Algorithm for strict_output function (gpt.ts file)

Below is the pseudocode for working of the strict_output function.

```
FUNCTION strict_output(
  system_prompt: STRING,
  user_prompt: STRING or STRING ARRAY,
  output_format: OBJECT,
  default_category: STRING OPTIONAL,
  output_value_only: BOOLEAN,
  model: STRING = "gpt-4o mini",
  temperature: NUMBER = 1,
  num_tries: NUMBER = 3,
  verbose: BOOLEAN = FALSE
)
  list_input ← (user_prompt IS ARRAY)
  dynamic_elements ← ('<' AND '>' IN output_format)
  list_output ← ('[' AND ']' IN output_format)
  error_msg ← ""

  FOR i FROM 1 TO num_tries
    output_format_prompt ←
BUILD_OUTPUT_FORMAT_PROMPT(output_format)
    system_message ← system_prompt +
output_format_prompt + error_msg
    user_message ← CONVERT_TO_STRING(user_prompt)

    response ← CALL_OPENAI_API(model, temperature,
system_message, user_message)
    response ← REPLACE_SINGLE_QUOTES(response)

    IF verbose THEN
      LOG(system_prompt, user_message, response)

    IF VALIDATE_OUTPUT(response, list_input,
output_format, default_category, output_value_only) THEN
      RETURN PARSED_OUTPUT

    error_msg ← UPDATE_ERROR_MSG(response,
error_msg)

  RETURN EMPTY_ARRAY

FUNCTION
BUILD_OUTPUT_FORMAT_PROMPT(output_format)
  // build output_format_prompt string explaining desired
output format to GPT-4o mini

FUNCTION CALL_OPENAI_API(model, temperature,
system_message, user_message)
  // send request to OpenAI API using createChatCompletion

FUNCTION REPLACE_SINGLE_QUOTES(response)
  // replace single quotes with double quotes in response

FUNCTION VALIDATE_OUTPUT(response, list_input,
output_format, default_category, output_value_only)
  // try parsing response as JSON
  parsed_output ← PARSE_JSON(response)

  IF list_input THEN
    parsed_output ← WRAP_IN_ARRAY(parsed_output)

  FOR EACH element IN parsed_output
    FOR EACH key IN output_format
      // skip keys containing '< >' (dynamic elements)
      IF key CONTAINS '< >' THEN
        CONTINUE

      // ensure key exists in output element
      IF key NOT IN element THEN
        RAISE ERROR

      // validate key value based on output_format
      IF key SPECIFIES LIST THEN
        // ensure output value is not an array itself (take first
element)
        // use default_category if GPT-4o mini couldn't
identify a category
        // extract only label before colon if format is
description

      IF output_value_only THEN
        // convert output element to array of values
        // if array contains only one element, return that
element directly
  RETURN TRUE

FUNCTION UPDATE_ERROR_MSG(response, error_msg)
  // update error_msg with response and error message
```

Algorithm for POST function for creating course outline (route.ts file) for /api/course/createChapters endpoint:

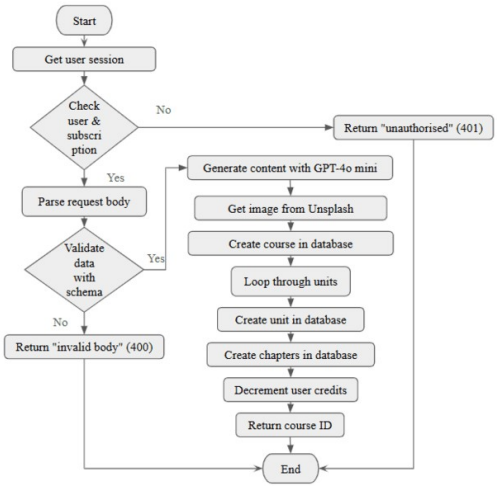

Fig 5.2.2. Algorithm for POST function for creating course outline (route.ts file)

Below is the pseudocode for working of the POST function for creating course outline.

```
FUNCTION POST(req: REQUEST_OBJECT)
  // Authentication and Authorization
  session_info ← getAuthSession(req)
  IF session_info IS NULL THEN
    RETURN ERROR_RESPONSE(401, "unauthorized")
  END IF

  subscription_status ← checkSubscription(session_info)
  IF subscription_status IS "zero_credits" AND NOT session_info.pro_member THEN
    RETURN ERROR_RESPONSE(402, "no credits")
  END IF

  // Data Parsing and Validation
  request_body ← req.json()
  title, units ← createChaptersSchema.parse(request_body)

  // Content Generation with GPT-4o mini
  output ← strict_output(
    "Generate chapter content for each unit, including titles and Youtube search queries.",
    units,
    // output format object
  )
  image_search_term ← strict_output(
    "Generate an image search term for the course title.",
    title,
    // output format object
  )

  // Image Retrieval
  image_url ← getUnsplashImage(image_search_term)

  // Database Interactions (Prisma)
  course_id ← prisma.course.create({ title, image_url })
  FOR EACH unit IN output.units
    unit_id ← prisma.unit.create({ course_id, unit })
    prisma.chapter.createMany({ unit_id, chapters: unit.chapters })
  END FOR

  // Decrement User Credits
  prisma.user.update({ id: session_info.user_id, credits: session_info.credits - 1 })

  // Return Response
  RETURN NEXT_RESPONSE({ course_id })

  // Error Handling
  CATCH error
    IF error IS ZodValidationError THEN
      RETURN ERROR_RESPONSE(400, "invalid body")
    ELSE
      RETURN ERROR_RESPONSE(500, "internal server error")
    END IF
  END CATCH
```

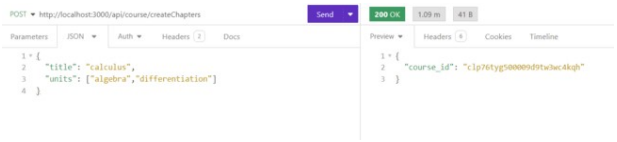

Fig 5.2.3. Testing API endpoint on Insomnia

Above figure is a screenshot representing the Testing of API endpoint on Insomnia application with input as title and units shown on left hand side and course id output shown on right hand side in json format.

Algorithm for POST function (route.ts) for /api/chapter/getInfo endpoint:

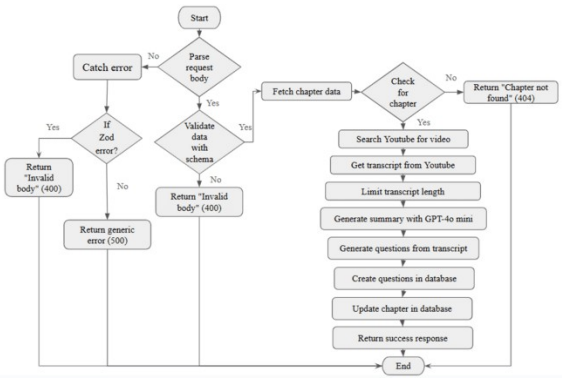

Fig 5.2.4. Algorithm for POST function (route.ts) for /api/chapter/getInfo endpoint

Below is the pseudocode for working of the POST function (route.ts) for /api/chapter/getInfo endpoint.

```
FUNCTION POST(req: REQUEST_OBJECT)
  // Data Parsing and Validation
  request_body ← req.json()
  IF NOT bodyParser.parse(request_body, "chapterId") THEN
    RETURN ERROR_RESPONSE(400, "invalid body")
  END IF

  chapterId ← request_body.chapterId

  // Fetch Chapter Data
```

```
  chapter_data ← prisma.chapter.findUnique({ id: chapterId })
  IF chapter_data IS NULL THEN
    RETURN ERROR_RESPONSE(404, "Chapter not found")
  END IF

  // Content Retrieval and Processing
  videoId ← searchYoutube(chapter_data.youtubeSearchQuery)
  IF videoId IS NULL THEN
    // handle video search error
  END IF

  transcript ← getTranscript(videoId)
  IF transcript IS NULL THEN
    // handle transcript fetch error
  END IF

  transcript ← LIMIT_TRANSCRIPT_LENGTH(transcript, 500)

  // Summary Generation with GPT-4o mini
  summary ← strict_output(
    "Summarize the transcript in 250 words or less, excluding irrelevant information.",
    transcript,
    // output format object
  )

  // Question Generation
  questions ← getQuestionsFromTranscript(transcript, chapter_data.name)

  // Database Interactions (Prisma)
  prisma.question.createMany(questions)
  prisma.chapter.update({ id: chapterId, videoId, summary })

  // Return Response
  RETURN NEXT_RESPONSE({ success: true })

  // Error Handling
  CATCH error
    IF error IS ZodValidationError THEN
      RETURN ERROR_RESPONSE(400, "invalid body")
    ELSE
      RETURN ERROR_RESPONSE(500, "internal server error", error)
    END IF
  END CATCH
```

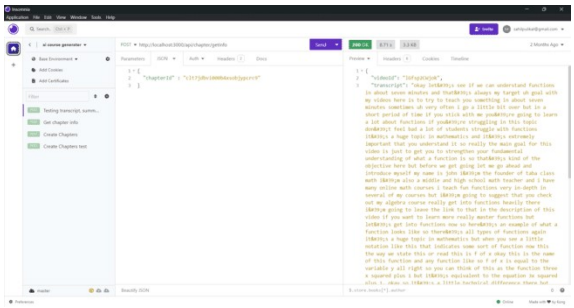

Fig 5.2.5. Testing API endpoint on Insomnia

Above figure is a screenshot representing the Testing of API endpoint on Insomnia application with chapterID as input shown on left and output shown on right side containing videoId and transcript in json format.

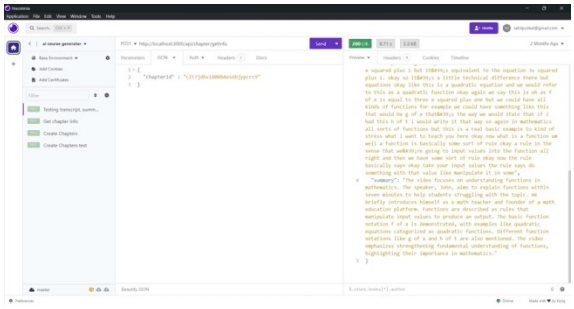

Fig 5.2.6. Testing of API endpoint done using Insomnia

Above screenshot is the continuation of Fig 5.2.5, showing the generated summary in json format on the right hand side.

3. Embedding YouTube videos:

The application is enhanced by integrating the YouTube API to enrich course materials and provide a comprehensive learning experience.

An API key for YouTube is obtained by enabling the YouTube Data API v3 and creating a Google Cloud Platform account, allowing for authorized requests to be sent to the API. A new YouTube client is launched after installing the YouTube Data API Node.js SDK, facilitating API interactions.

Pertinent videos are searched for using the YouTube API during course creation, retrieving video titles, descriptions, and thumbnails to be incorporated into the course materials.

Relevant YouTube videos are embedded directly into the course material, augmenting the course content and providing a more engaging learning experience for students.

Interactive tests are generated using video material, leveraging the YouTube API to create multiple-choice questions based on the embedded videos' transcript material, enabling students to assess their understanding and reinforce key concepts.

Algorithm for youtube.ts functions:

1. searchYoutube(searchQuery: string):

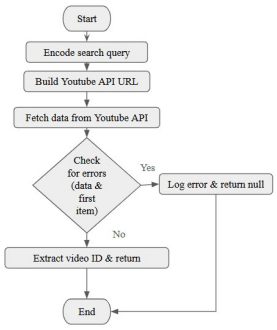

Fig 5.2.7. Algorithm for searchYoutube function

Below is the pseudocode for working of the searchYoutube function.

```
FUNCTION searchYoutube(searchQuery: STRING)
  // Encode search query
  encoded_query ← encodeURIComponent(searchQuery)

  // Build Youtube API request URL
  url ← "https://www.googleapis.com/youtube/v3/search"
  params ← {
    key: process.env.YOUTUBE_API_KEY,
    q: encoded_query,
    videoDuration: "medium",
    videoEmbeddable: true,
    type: "video",
    maxResults: 5
  }

  // Fetch data from Youtube API
  response ← axios.get(url, params)
  data ← response.data

  // Error handling
  IF data IS NULL THEN
    LOG_ERROR("No data returned from Youtube API")
    RETURN NULL
  END IF

  IF data.items[0] IS UNDEFINED THEN
    LOG_ERROR("No videos found for search query")
    RETURN NULL
  END IF

  // Extract video ID
  videoId ← data.items[0].id.videoId
  RETURN videoId
```

2. getTranscript(videoId: string):

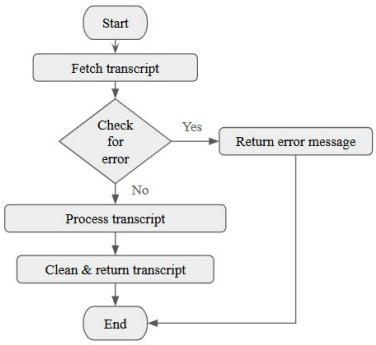

Fig 5.2.8. Algorithm for getTranscript function

Below is the pseudocode for working of the getTranscript function.

```
FUNCTION getTranscript(videoId: STRING)
  // Fetch transcript
transcript_array←
YoutubeTranscript.fetchTranscript(videoId, "en")
  IF transcript_array IS NULL THEN
    RETURN "Error fetching transcript"
  END IF

  // Process transcript
  transcript ← ""
  FOR EACH t IN transcript_array
    transcript ← transcript + t.text + " "
  END FOR

  // Clean and return transcript
  transcript ← transcript.replaceAll("\n", "")
  RETURN transcript

  // Error handling
  CATCH error
    RETURN "Error: " + error
  END CATCH
```

3. getQuestionsFromTranscript(transcript: string, course_title: string):

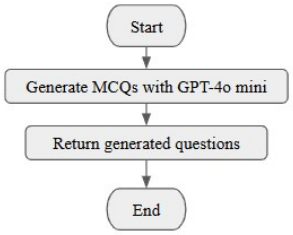

Fig 5.2.9. Algorithm for getQuestionsFromTranscript function

Below is the pseudocode for working of the getQuestionsFromTranscript function.

```
FUNCTION getQuestionsFromTranscript(transcript:
STRING, course_title: STRING)
  // Generate multiple choice questions
  prompts ← new Array(5).fill({
    prompt: "Generate a random hard multiple-choice
question with context from the transcript.",
    transcript: transcript,
    course_title: course_title
  })
  questions ← strict_output(prompts, {
    output_format: {
      question: STRING,
      answer: STRING,
      options: ARRAY OF STRING
    }
  })

  // Return generated questions
  RETURN questions
```

4. Incorporating Stripe to enable paid features for users.

To integrate Stripe payments, obtaining an API key and setting up a Stripe account are necessary steps.

A Stripe account is created and an API key is obtained by signing up for a free account on the Stripe website.

The Stripe Node.js SDK is installed immediately after receiving the API key, followed by the initialization of a new Stripe client.

The application framework utilizes the API key to integrate Stripe payments, enabling secure and efficient transaction processing.

Algorithm for Handling Stripe Payments (GET request, route.ts)

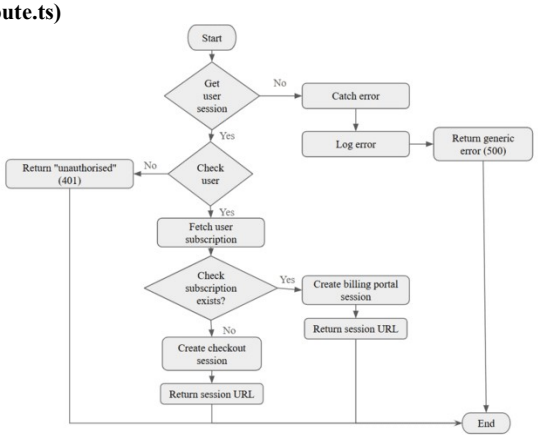

Fig 5.2.10. Algorithm for Handling Stripe Payments

Below is the pseudocode for working of the Algorithm for Handling Stripe Payments.

```
FUNCTION HANDLE_STRIPE_PAYMENTS()
  // Authentication and User Check
  session_info ← getAuthSession()
  IF session_info IS NULL THEN
    RETURN ERROR_RESPONSE(401, "unauthorized")
  END IF

  // Fetch User Subscription
  user_subscription ← prisma.userSubscription.findUnique({
userId: session_info.userId })
  IF user_subscription IS NOT NULL THEN
    // Existing Subscription Handling
    stripe_session ← stripe.billingPortal.sessions.create({
customer: user_subscription.stripeCustomerId })
    RETURN NEXT_RESPONSE({ url: stripe_session.url })
  END IF

  // New Subscription Creation
  stripe_session ← stripe.checkout.sessions.create({
    success_url: "/settings",
    cancel_url: "/settings",
    payment_method_types: ["card"],
    subscription_mode: true,
    billing_address_collection: true,
    customer_email: session_info.email,
    line_items: [{
      price_data: {
        currency: "inr",
        product_data: {
          name: "Subscription",
          description: "Monthly subscription"
        },
        unit_amount: 1000
      },
      recurring: {
        interval: "month"
      },
      quantity: 1,
      metadata: {
        userId: session_info.userId
      }
    }]
  })
  RETURN NEXT_RESPONSE({ url: stripe_session.url })

  // Error Handling
  CATCH error
    LOG_ERROR(error)
    RETURN ERROR_RESPONSE(500, "internal server
error")
  END CATCH
```

Algorithm for Stripe Webhook (POST request, route.ts)

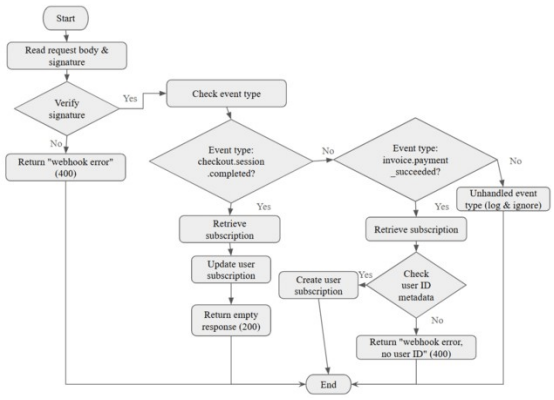

Fig 5.2.10. Algorithm for Stripe Webhook

Below is the pseudocode for working of the Algorithm for Stripe Webhook.

```
FUNCTION STRIPE_WEBHOOK(req:
REQUEST_OBJECT)
  // Parse Request Body and Signature
  body ← req.text()
  signature ← req.headers["Stripe-Signature"]

  // Verify Signature
  TRY
    event ← stripe.webhooks.constructEvent(body,
signature, WEBHOOK_SECRET)
  CATCH error
    RETURN ERROR_RESPONSE(400, "webhook error")
  END TRY

  // Process Stripe Event
  SWITCH event.type
    CASE "checkout.session.completed"
      session ←
stripe.checkout.sessions.retrieve(event.data.object.id)
      subscription ←
stripe.subscriptions.retrieve(session.subscription)
      IF session.metadata.userId IS NULL THEN
        RETURN ERROR_RESPONSE(400, "webhook
error, no user id")
      END IF
      prisma.userSubscription.create({
        userId: session.metadata.userId,
        stripeSubscriptionId: subscription.id,
        stripeCustomerId: subscription.customer,
        stripePriceId: subscription.items.data[0].price.id,
        stripeCurrentPeriodEnd:
subscription.current_period_end
      })
    CASE "invoice.payment_succeeded"
```

```
        // similar processing as above
    END SWITCH

    // Return empty response with 200 status code
    RETURN NEXT_RESPONSE({}, 200)
```

5. Prompt Engineering Techniques

Prompt engineering, the art of crafting effective prompts for large language models, is crucial in developing sophisticated applications like course generator applications. By meticulously designing prompts, developers can harness the power of AI to produce high-quality, tailored content. This section delves into the core prompt engineering techniques employed, focusing on their role in guiding the large language model towards generating desired course content.

Key prompt engineering techniques used:

Role-Based Prompting: Defining the AI's role or persona to align its output with the desired task.

Task-Specific Instructions: Clearly outlining the desired outcome and providing specific guidelines.

Output Formatting: Specifying the expected output structure to ensure consistency and compatibility.

Iterative Prompting: Employing multiple prompts or refining prompts based on initial results to improve output quality.

Constraint Enforcement: Explicitly defining limitations or requirements to guide the model's response.

By effectively combining these techniques, the prompts successfully guide the large language model to generate relevant course content, including chapter titles, YouTube search queries, and multiple-choice questions. This demonstrates the potential of prompt engineering to enhance the capabilities of this AI-powered application.

Prompt Engineering for Course Outline Creation

```
//prompt engineering
let output_units: outputUnits = await strict_output(
  "You are an AI capable of curating course content, coming up with relevant chapter titles, and finding relevant
  youtube videos for each chapter",
  new Array(units.length).fill(
    `It is your job to create a course about ${title}. The user has requested to create chapters for each of the
    units ${units.join(
      ", "
    )}. Each unit should contain more than one relevant chapers related to that particular unit and ${title}. Then,
    for each chapter, provide a detailed youtube search query that can be used to find an informative educational
    video for each chapter. Each query should give an educational informative course in youtube.`
  ),
  {
    title: "string",
    chapters: [
      {
        chapter_title: "string",
        youtube_search_query: "string",
      },
    ],
  }
);
```

Fig 5.2.11. Actual code utilizing Prompt Engineering for Course Outline Creation

Role-Based Prompting: The initial prompt establishes the AI model's role as a "course content curator," setting the context for the subsequent task. This technique helps to align the model's output with the desired goal.

Task-Specific Instructions: The prompt clearly outlines the task, including generating chapter titles, finding relevant YouTube videos, and structuring the output in a specific format. This provides clear guidelines for the model to follow.

Output Formatting: The output_format parameter explicitly defines the expected JSON structure of the output. This acts as a template for the model, guiding it to generate content in the desired format.

Iterative Prompting: It is used to improve the quality of the generated output over multiple attempts.

Prompt Engineering Best Practices: The code adheres to best practices by providing clear and concise instructions, avoiding ambiguity, and using consistent language.

Contextual Understanding: The prompt leverages the provided course title and units to provide relevant context for the AI model. This helps in generating more accurate and relevant content.

Output Control: The output_format parameter acts as a control mechanism to ensure the generated output adheres to the desired structure.

Prompt Engineering for Image search term for Unsplash

```
const imageSearchTerm = await strict_output(
  "you are an AI capable of finding the most relevant image for a course",
  `Please provide a good image search term for the title of a course about ${title}. This search term will be fed
  into the unsplash API, so make sure it is a good search term that will return good results`,
  {
    image_search_term: "a good search term for the title of the course",
  }
);
```

Fig 5.2.12. Actual code utilizing Prompt Engineering for Image search term for Unsplash

Role-Based Prompting: The prompt explicitly defines the AI's role as an expert in finding relevant images for courses. This helps align the model's response with the desired task.

Task-Specific Instructions:

- Image Search Term Generation: The prompt clearly outlines the task of generating a suitable image search term based on the provided course title.
- API Consideration: The prompt emphasizes the importance of generating a search term that will yield effective results on the Unsplash API, indicating a practical understanding of the downstream application.

Explicit Output Structure Formatting: The output_format parameter dictates the desired output structure, ensuring the model generates a single string as the image search term.

Prompt Engineering for YouTube Transcript summarization

```
const { summary }: { summary: string } = await strict_output(
  "You are an AI capable of summarising  a youtube transcript",
  "summarise in 250 words or less and do not talk about the sponsors or anything unrelated to the main topic, also
  do not introduce what the summary is about. And do not include special characters, back slashes, and do not
  highlight any text using double quotation marks symbol and proper json format is required strictly.\n" +
    transcript,
  { summary: "summary of the transcript" }
);
```

Fig 5.2.13. Actual code utilizing Prompt Engineering for YouTube Transcript summarization

Role-Based Prompting: The initial prompt clearly defines the AI's role as a summarizer, setting the stage for the subsequent task.

Task-Specific Instructions: The prompt explicitly outlines the summarization task, including word count limits, exclusion criteria (sponsors, unrelated content), and formatting requirements.

Output Formatting: The output_format parameter specifies the desired output structure, ensuring a clean JSON output.

Constraint and Format Enforcement: The prompt emphasizes the avoidance of special characters, backslashes, and quotation marks, ensuring a well-structured and compatible output. The strict requirement for JSON format ensures compatibility with downstream processes.

Clarity and Conciseness: The prompt is clear and concise, directly conveying the desired outcome.

Specificity: The prompt provides specific guidelines regarding word count and content exclusion, enhancing the quality of the generated summary.

Prompt Engineering for Quiz Generation

```
const questions: Question[] = await strict_output(
  "You are a helpful AI that is able to generate mcq questions and answers, the length of each answer should not be
  more than 15 words. And do not include special characters, back slashes, and do not highlight any text using double
  quotation marks symbol and proper json format is required strictly.",
  new Array(5).fill(
    `You are to generate a random hard mcq question about ${course_title} related to ${searchQuery} with context of
    the following transcript: ${transcript}`
  ),
  {
    question: "question",
    answer: "answer with max length of 15 words",
    option1: "option1 with max length of 15 words",
    option2: "option2 with max length of 15 words",
    option3: "option3 with max length of 15 words",
  }
);
return questions;
```

Fig 5.2.14. Actual code utilizing Prompt Engineering for Quiz Generation

Role-Based Prompting: The initial prompt explicitly defines the AI's role as a "helpful AI" capable of generating MCQs, setting clear expectations for the model's output.

Task-Specific Instructions: The prompt clearly outlines the task, including the desired format (MCQs with specific length constraints), the difficulty level (hard), and the requirement to generate questions based on the provided transcript and course title.

Output Formatting: The output_format parameter explicitly defines the expected JSON structure of the output, ensuring consistency and ease of processing.

Iterative Prompting: The use of an array to generate multiple questions demonstrates a basic form of iterative prompting, allowing for the generation of a set of questions based on the same core prompt.

Constraints and Guidelines: The prompt explicitly states requirements for answer length, special character avoidance, and JSON formatting, providing clear guidelines for the model.

Contextual Information: The prompt incorporates the course_title and transcript into the question generation process, providing essential context for the model.

Difficulty Level Specification: The prompt specifies the desired difficulty level of the questions (hard), influencing the model's output.

6. Setup the application in a cloud production environment.

The final stages involve building the application and deploying it on a live Virtual Private Server (VPS).

The application framework is built and prepared for production deployment.

Numerous hosting companies are available for deployment, with options including the Google Cloud Platform, among others.

The application is deployed to a live VPS, ensuring a secure and scalable environment for users to access and utilize the application.

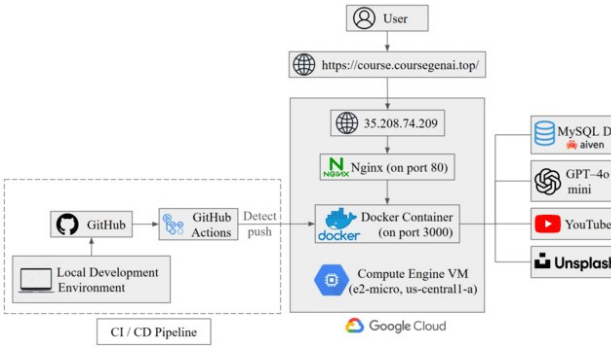

Fig 5.2.15. Application architecture diagram

This application architecture employs an intricate CI/CD pipeline deeply integrated with resilient cloud infrastructure to ensure an optimized user experience. Development and testing of the code are conducted in isolated local environments before being consolidated into a central GitHub repository. Upon detecting any push to this repository, GitHub Actions are activated to orchestrate sophisticated workflows such as linting, testing, and Docker image construction, thereby streamlining and automating the deployment process with high efficiency.

The backend of the application is deployed on a Google Cloud Compute Engine VM, specifically an e2-micro instance situated in the us-central1-a region, providing a highly scalable and reliable computing environment. The application itself operates within a Docker container, ensuring uniform performance across diverse environments, with the container actively listening on port 3000. An Nginx server functions as a reverse proxy, handling traffic on port 80 and seamlessly forwarding incoming requests to the Docker container. This setup not only abstracts complexity but also enhances security and performance. The application is consistently reachable via an external IP address, 35.208.74.209, ensuring stable and persistent connectivity.

To augment functionality, the application integrates with multiple external services. A MySQL database hosted on Aiven is used to securely store critical application data, including user profiles, course metadata, and video information. OpenAI's GPT-4o mini API is utilized to dynamically generate course content such as chapter titles and detailed descriptions, leveraging state-of-the-art language processing capabilities. The YouTube API is employed to retrieve relevant video IDs and transcripts, enabling the application to provide curated video content for each course. Additionally, the Unsplash API is integrated to source visually compelling images that enhance the course materials.

Users engage with the application via a sophisticated web interface accessible at https://course.coursegenai.top/. They commence by signing in through Google authentication. To generate a course, users input the course title and subtopics, after which the system processes this input and interacts with various APIs to produce comprehensive course content. Users can review and modify the generated chapters before finalizing the course. The application systematically retrieves the necessary images, videos, and transcripts from the corresponding APIs, ensuring that all multimedia content is highly relevant and of superior quality.

The final course content, together with its multimedia elements, is stored in the MySQL database and presented on the frontend, delivering a rich and interactive educational experience to the users. This architecture, with its sophisticated combination of cloud computing, containerization, and extensive API integrations, provides a highly efficient, seamless, and user-friendly course generation platform, embodying advanced technological capabilities.

6. RESULTS & DISCUSSIONS

6.1 Testing and comparing output with ChatGPT output and our application output:

Course outline generation using ChatGPT 3.5:

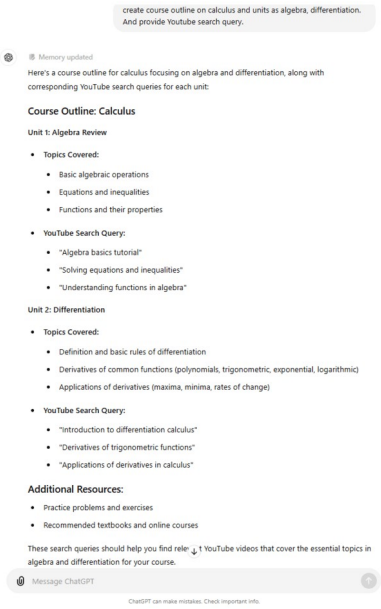

Fig. 6.1.1 Course outline generation using ChatGPT 3.5

ChatGPT generated a response that, while generally accurate, included extraneous information such as additional resources at the end that are not pertinent to our specific needs. The response would benefit from a more focused and precise output tailored to our requirements.

Course outline generation using our Application, testing done using Insomnia for testing our API endpoint:

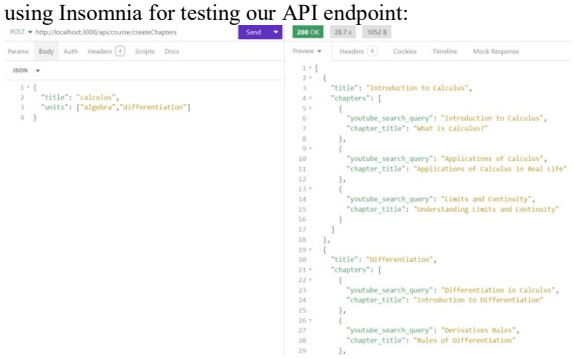

Fig. 6.1.2 Course outline generation using our Application with input on left and output on right.

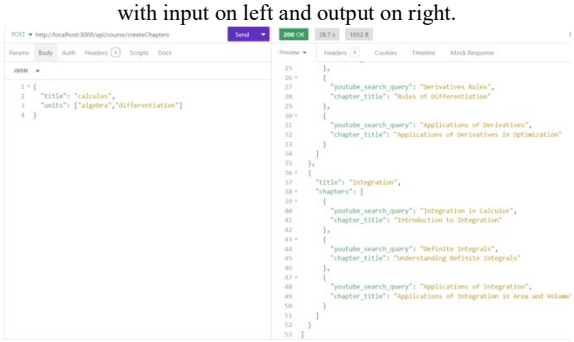

Fig. 6.1.3 Course outline generation using our Application with input on left and output on right.

In the above illustrated example, it is evident that our application generates meticulously structured JSON responses, adhering strictly to the specified schema. These responses are precise and exclusively contain the required data elements, devoid of any extraneous information. This precision facilitates seamless data parsing and accurate display on the front end while ensuring efficient and error-free integration with the backend database systems.

This characteristic offers several advantages in the context of front-end interaction and database persistence:

Enhanced Parsability: Precise JSON structures adhere to a well-defined syntax, enabling efficient parsing by front-end framework. This minimizes parsing errors and streamlines data manipulation within the user interface.

Data Precision: Conveying only the necessary information within the JSON responses ensures clarity and eliminates the need for extraneous data filtering or transformation on the front-end. This reduces processing overhead and improves application performance.

Simplified Database Integration: The well-defined schema inherent in JSON aligns well with relational database structures. This facilitates seamless data persistence with minimal mapping or schema conversion efforts, leading to a more robust and maintainable data layer.

Reduced Coupling: By adhering to a standardized JSON format, the application establishes a loose coupling between the front-end and back-end. This allows for independent development and maintenance of each layer, fostering greater flexibility and scalability.

In conclusion, the application's focus on well-formed JSON responses fosters efficient data exchange, simplifies front-end development, and promotes a clean and maintainable architecture. This approach lays the groundwork for a robust and scalable application ecosystem.

YouTube Transcript summarization queried on ChatGPT 3.5:

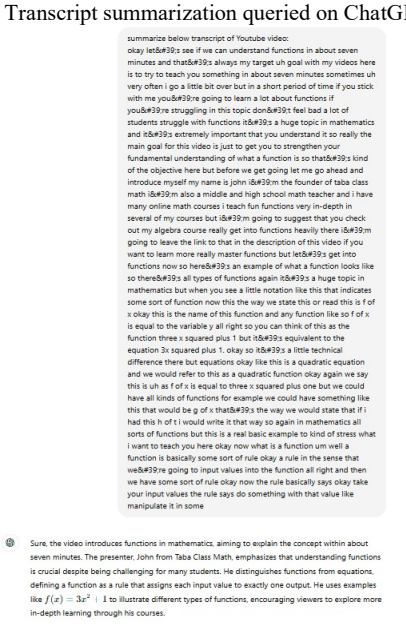

Fig. 6.1.4 Summary of video transcript from ChatGPT 3.5

In the aforementioned example, ChatGPT exhibits a limited capacity, restricted to generating only a summary.

YouTube Transcript summarization using our Application, testing done using Insomnia for testing our API endpoint:

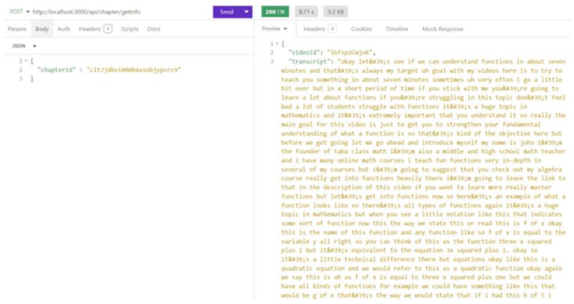

Fig. 6.1.5 YouTube video id and Transcript fetching using our Application, testing done using Insomnia for testing our API endpoint

Fig. 6.1.6 YouTube Transcript summarization using our Application, testing done using Insomnia for testing our API endpoint

In the above example, our application demonstrates the capability to generate relevant YouTube video IDs alongside their corresponding transcripts, leveraging the YouTube Data API v3 for a multi-faceted approach to content acquisition and processing. Furthermore, it synthesizes comprehensive summaries from these transcripts. This entire process is executed simultaneously, delivering the output in a meticulously structured JSON format. Such structured output ensures seamless integration with both the frontend display mechanisms and backend database systems. This not only enhances data parsing and presentation but also streamlines the end-user experience, rendering it seamless, efficient, and user-friendly.

Targeted Video Selection: By seamlessly extracting relevant YouTube video IDs, the application employs the YouTube Data API v3 for efficient content discovery. This ensures that only videos pertinent to the user's needs are retrieved, minimizing extraneous data and improving information retrieval accuracy.

Transcript Acquisition and Summarization: Building upon the video ID, the application retrieves transcripts through the YouTube Data API v3. Subsequently, it utilizes prompt engineering techniques for text summarization using GPT 4o mini to condense the transcripts into concise summaries. This process not only provides the user with key takeaways from the video but also minimizes data volume for front-end rendering and database storage, optimizing resource utilization.

Well-Structured JSON Formatting: The application meticulously organizes the extracted video ID, transcript, and summary within a well-defined JSON structure. This standardized format facilitates seamless data consumption by the front-end framework. It further streamlines database integration, enabling efficient storage and retrieval of the processed content.

In essence, this approach fosters a cohesive workflow for content acquisition, processing, and delivery. The targeted selection of videos, combined with the generation of concise summaries, ensures a streamlined user experience. The well-structured JSON output simplifies data exchange and integration across various application layers, resulting in a robust and scalable architecture.

6.2 Actual snapshots of application:

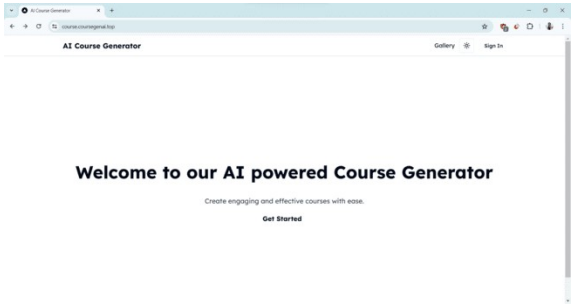

Fig. 6.2.1. Landing page

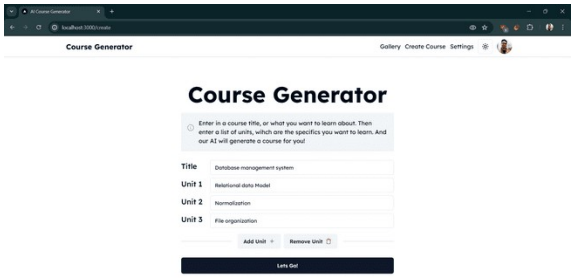

Fig. 6.2.2. User enters the appropriate required course units and Title.

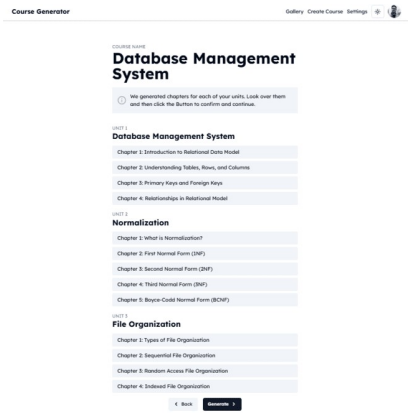

Fig. 6.2.3. After reviewing the course outline they generated, the user selects the “Generate” option.

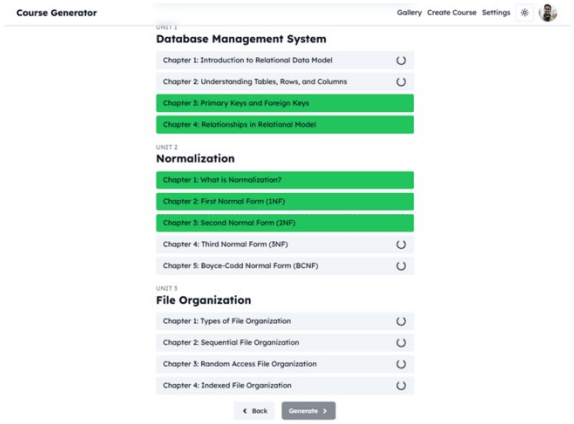

Fig. 6.2.4. Boxes turn green after Asynchronously retrieving pertinent YouTube videos along with transcripts, then producing a transcript summary.

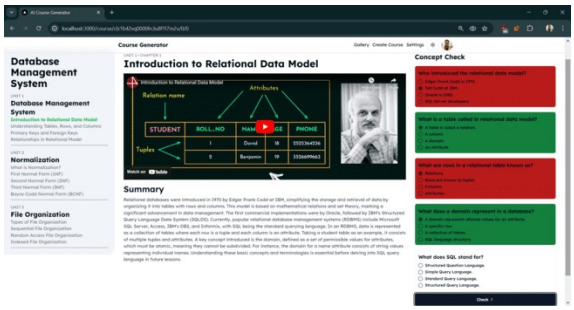

Fig. 6.2.5. The final course page includes chapters and units created by AI, as well as quizzes, related YouTube videos, and summaries of the videos. Correct and erroneous answers are highlighted in green and red in this AI-generated summary derived from the transcripts of YouTube videos and quizzes.

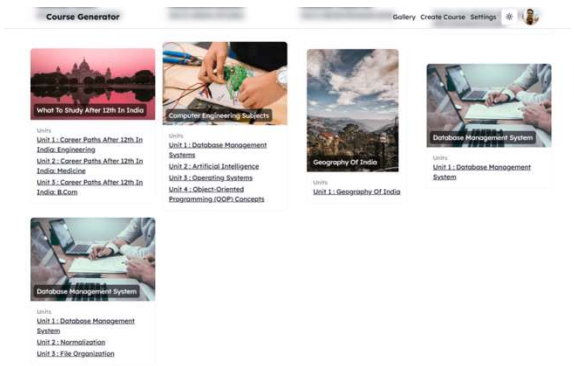

Fig. 6.2.6. The produced courses on the Gallery Page have pertinent photos from Unsplash.

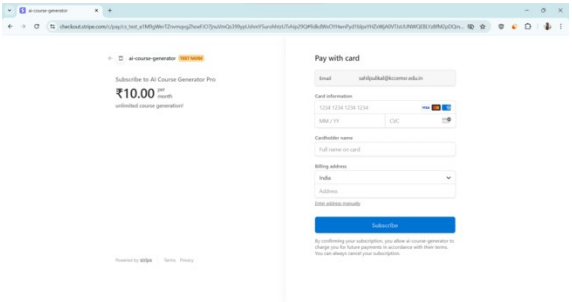

Fig. 6.2.7. The user can input their credit or debit card information on the Payment page after selecting Upgrade.

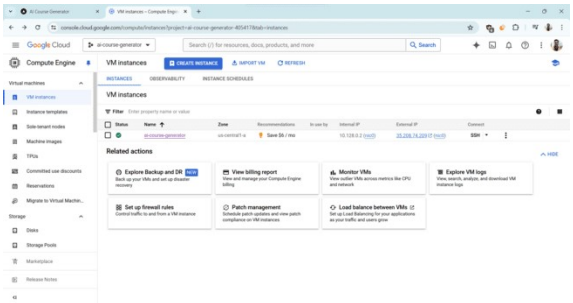

Fig. 6.2.8. Deployed the application on a Google Cloud Compute Engine virtual machine for scalable and secure hosting.

7. CONCLUSION AND FUTURE SCOPE

This innovative course creation platform leverages generative artificial intelligence (AI) to empower users to design highly customized courses on various topics, complete with interactive assessments and YouTube video integration, transcending the limitations of traditional applications.

By harnessing the power of AI, this tool streamlines the course creation process, saving users time and effort by generating quizzes, finding relevant YouTube content, developing outlines, and summarizing videos. It ensures tailored outlines, offers limitless course creation through Stripe payments integration, and features a user-friendly course gallery with visually appealing thumbnails courtesy of the Unsplash API. The latest OpenAI GPT 4o mini API and advanced prompts generate a wealth of engaging content, while customized quizzes, summaries, and concept identification enhance the learning experience.

With vast potential to revolutionize education, this platform has exciting future possibilities:

Cooperative Course Creation: Introduce features enabling users to collaborate on course creation and sharing, fostering a more dynamic educational environment.

LMS Integration: Develop tools for seamless import and export of created courses, facilitating integration into existing educational platforms, and exploring connections with online assessment tools.

Multimodal Learning: Consider incorporating additional modalities like audio and images to create a more comprehensive and engaging learning experience.

Multilingual Support: Incorporate language support to break down educational barriers and expand the platform's reach.

By continuously improving and updating this Generative AI-powered course creation platform, we aim to democratize education, providing access to creative and stimulating learning experiences for educators and learners alike.

8. REFERENCES

[1] R. AlAli and Y. Wardat, "Opportunities and Challenges of Integrating Generative Artificial Intelligence in Education," International Journal of Religion, vol. 5, no. 7, pp. 784–793, May 2024, doi: https://doi.org/10.61707/8y29gv34.

[2] R. Kaplan-Rakowski, K. Grotewold, P. Hartwick, and K. Papin, "Generative AI and Teachers' Perspectives on Its Implementation in Education," Journal of Interactive Learning Research, vol. 34, no. 2, pp. 313–338, 2023, Available: https://www.learntechlib.org/p/222363/

[3] A. Ghimire, J. Prather, and J. Edwards, "Generative AI in Education: A Study of Educators' Awareness, Sentiments, and Influencing Factors," arXiv.org, Mar. 22, 2024. https://arxiv.org/abs/2403.15586

[4] M. Alier, Francisco-José García-Peñalvo, and J. D. Camba, "Generative Artificial Intelligence in Education: From Deceptive to Disruptive," International Journal of Interactive Multimedia and Artificial Intelligence, vol. 8, no. 5, pp. 5–5, Jan. 2024, doi: https://doi.org/10.9781/ijimai.2024.02.011.

[5] Thomas, "Future research recommendations for transforming higher education with generative AI," Computers & Education: Artificial Intelligence, vol. 6, pp. 100197–100197, Jun. 2024, doi: https://doi.org/10.1016/j.caeai.2023.100197.

[6] R. AlAli, Y. Wardat, K. Al-Saud, and K. A. Alhayek, "Generative AI in Education: Best Practices for Successful Implementation," International Journal of Religion, vol. 5, no. 9, pp. 1016–1025, Jun. 2024, doi: https://doi.org/10.61707/pkwb8402.

[7] L. Bonde, "A Conceptual Design of a Generative Artificial Intelligence System for Education," International journal of research and innovation in applied science, vol. IX, no. IV, pp. 457–469, Jan. 2024, doi: https://doi.org/10.51584/ijrias.2024.904034.

[8] D. Griffiths, E. Frías-Martínez, A. Tlili, and D. Burgos, "A Cybernetic Perspective on Generative AI in Education: From Transmission to Coordination," International journal of interactive multimedia and artificial intelligence, vol. In press, no. In press, pp. 1–1, Jan. 2024, doi: https://doi.org/10.9781/ijimai.2024.02.008.

[9] A. Alammari, "Evaluating generative AI integration in Saudi Arabian education: a mixed-methods study," PeerJ Computer Science, vol. 10, p. e1879, Feb. 2024, doi: https://doi.org/10.7717/peerj-cs.1879.

[10] N. McDonald, A. Johri, A. Ali, and A. Hingle, "Generative Artificial Intelligence in Higher Education: Evidence from an Analysis of Institutional Policies and Guidelines," arXiv (Cornell University), Jan. 2024, doi: https://doi.org/10.48550/arxiv.2402.01659.

[11] A. Ghimire, "Generative AI in Education From the Perspective of Students, Educators, and Administrators," All Graduate Theses and Dissertations, Fall 2023 to Present, May 2024, Available: https://digitalcommons.usu.edu/etd2023/124/

[12] D. Grover, "Next-Generation Education: The Impact of Generative AI on Learning," Jan. 2024, doi: https://doi.org/10.52783/jier.v4i2.1019.

[13] H. Li et al., "Bringing Generative AI to Adaptive Learning in Education," arXiv (Cornell University), June. 2024, doi: https://doi.org/10.48550/arxiv.2402.14601.

[14] M. Resnick, "Generative AI and Creative Learning: Concerns, Opportunities, and Choices," An MIT Exploration of Generative AI, Mar. 2024, doi: https://doi.org/10.21428/e4baedd9.cf3e35e5.

[15] G. Kurtz et al., "Strategies for Integrating Generative AI into Higher Education: Navigating Challenges and Leveraging Opportunities," Education Sciences, vol. 14, no. 5, p. 503, May 2024, doi: https://doi.org/10.3390/educsci14050503.

[16] D. E. Salinas-Navarro, E. Vilalta-Perdomo, R. Michel-Villarreal, and L. Montesinos, "Using Generative Artificial Intelligence Tools to Explain and Enhance Experiential Learning for Authentic Assessment," Education Sciences, vol. 14, no. 1, p. 83, Jan. 2024, doi: https://doi.org/10.3390/educsci14010083.

[17] I. Pesovski, R. Santos, R. Henriques, and V. Trajkovik, "Generative AI for Customizable Learning Experiences," Sustainability, vol. 16, no. 7, p. 3034, Jan. 2024, doi: https://doi.org/10.3390/su16073034.

[18] S. Ivanov, M. Soliman, A. Tuomi, Nasser Alhamar Alkathiri, and A. N. Al-Alawi, "Drivers of generative AI adoption in higher education through the lens of the theory of planned behaviour," Technology in Society, vol. 77, pp. 102521–102521, Mar. 2024, doi: https://doi.org/10.1016/j.techsoc.2024.102521.

[19] L.-H. Wong and Chee-Kit Looi, "Advancing the generative AI in education research agenda: Insights from the Asia-Pacific region," Asia Pacific Journal of Education, vol. 44, no. 1, pp. 1–7, Jan. 2024, doi: https://doi.org/10.1080/02188791.2024.2315704.

[20] R. Sajja, C. E. Ramirez, Z. Li, B. Z. Demiray, Y. Sermet, and I. Demir, "Integrating Generative AI in Hackathons: Opportunities, Challenges, and Educational Implications," arXiv.org, Feb. 01, 2024. https://arxiv.org/abs/2401.17434

[21] R. Gozalo-Brizuela and E. C. Garrido-Merchán, (2023) "A survey of Generative AI applications," arXiv.org, https://doi.org/10.48550/arXiv.2306.02781.

[22] R. Gozalo-Brizuela and E. C. Garrido-Merchan, (2023), "CHATGPT is not all you need. A state of the art review of large generative AI models," arXiv.org, https://doi.org/10.48550/arXiv.2301.04655.

[23] C. Zhang et al., (2023), "A complete survey on generative AI (AIGC): Is chatgpt from GPT-4 to GPT-5 all you need?," arXiv.org, https://doi.org/10.48550/arXiv.2303.11717.

[24] Su, J., & Yang , W. (2023). "Unlocking the Power of ChatGPT: A Framework for Applying Generative AI in Education. ECNU Review of Education", 6(3), 355-366. https://doi.org/10.1177/20965311231168423