INTEROPERABLE WEB SYSTEM FOR GENERATION OF LEARNING RESOURCES ON THE BASE OF LLM AND PROMPT ENGINEERING

ІНТЕРОПЕРАБЕЛЬНА ВЕБ-СИСТЕМА ДЛЯ СТВОРЕННЯ НАВЧАЛЬНИХ РЕСУРСІВ НА БАЗІ LLM ТА PROMPT ІНЖИНІРИНГУ

by Kakun Artem

Presented in Partial Fulfillment of the Requirements for the Degree

Master of Software Engineering

American University Kyiv

2024

APPROVED BY:

Sergiy Tytenko, Ph.D., Faculty Mentor

Abstract

The development of generative artificial intelligence (AI) and its various applications are still in the research and development phase. However, it is evident that the emergence of generative AI has a significant impact on various industries, including education.

This study explores the potential applications of generative AI in education, such as personalized learning tools and AI-powered study resources.

The work highlights the importance of prompt engineering in facilitating communication between users and AI systems, which can lead to better educational outcomes. Acknowledges the challenges and risks of incorporating AI technologies into education, such as data privacy and security concerns.

Acknowledgements

I express my deep gratitude to Dr. Sergiy Tytenko for his unwavering support and insightful guidance throughout my studies and the writing of this paper.

Furthermore, I am grateful to the American University Kyiv for providing me with the platform and opportunity to delve into this important topic as my Capstone Project.

Table of Contents

Acronyms table 4
Chapter 1. Generative AI and Prompt Engineering in an Educational Context 5
1.1. Generative AI in the Educational Process 5
1.2. Effective use of Prompt Engineering in the Educational Context 6
1.3. Chapter 1 Summary 8
Chapter 2. Capstone Project: AI-based Interoperable System for Generation of Learning Materials 9
2.1. Solution Overview 9
2.2. Requirements for AI-based Learning Materials Generation System 9
2.2.1. Business Value of Web System for Generation of Learning Materials 9
2.2.2. Architecturally Significant Requirements of the System 9
2.2.3. Non-functional requirements 10
2.3. Quality Attributes 11
2.3.1. Descriptions of Quality Attributes 11
2.4. Solution Architecture of the AI-based Learning Materials Generation Web System .. 12
2.4.1. AI-Based Web System Technology Stack Justification 12
2.4.2. Identified Project Risks 12
2.5. High-Level Solution Architecture of the AI-based Learning Materials Generation Web System 13
2.5.1. Context Diagram 13
2.5.2. Container Diagram 13
2.6. Implementation Details of the AI-based Learning Materials Generation Web System 14
2.6.1. AI-based Learning Materials Generation System Web Interface 14
2.6.2. Learning Material Generation Requests Processing 19
2.6.3. Trigger and Monitoring of the Materials Generation Process Within The System 22
2.6.4. Storing and Accessing Learning Materials Through Database Interaction 22
2.6.5. Integration With External LLM Services, Formation and Handling of the Generation Request 24
Chapter 3. Results and Analysis 25
3.1. Conclusions About the Web Interface and Backend Architecture 25
3.2. Conclusions About the Quality of Generated Materials 26
3.3. Possible Improvement Points 27
3.4. Results of the work done 27
References 29

Acronyms table

AI	Artificial Intelligence
API	Application Programming Interface
ASR	Architecturally Significant Requirement
DOM	Document Object Model
GPT	Generative Pre-trained Transformer
NLP	Natural Language Processing
NLG	Natural Language Generation
REST	Representational state transfer
UUID	A Universally Unique Identifier

Chapter 1. Generative AI and Prompt Engineering in an Educational Context

Generative AI is a powerful technology popularized by Chat GPT developed by Open AI. The Chat GPT website has 1.43 billion users as of August 2023 [1]. Chat GPT has brought the revolution to the public by demonstrating the ability to understand not only simple commands but also complex speech patterns and generate well-structured responses. Chat GPT is a product of deep learning [2], which is a subtype of machine learning that mirrors the human brain in learning and responding to data, information, and cues [3]. The big players in the tech world were quick to respond. Google announced its own generative AI - Bard.

Generative AI can be defined as a technology that (i) uses deep learning models to (ii) generate human-like content in response to (iii) complex and varied prompts (e.g., languages, instructions, questions) [18]. Several groundbreaking techniques have contributed to the development of generative AI, such as Generative Adversarial Networks (GANs) [5][6]. Eventually, the development of large-scale language models like BERT [7] and GPT [8] paved the way for the practical implementation of generative AI in various industries, including education.

It is important to distinguish generative AI from conversational AI. As defined above, generative AI has the ability to not only provide a response but also generate its content based on the information it has been trained on [4], which is a very big difference from conversational AI, which typically relies on pre-formed responses.

Key components of conversational AI include natural language processing (NLP), Natural Language Understanding (NLU), and natural language generation (NLG). It should be noted that not all generative AIs are conversational, nor do all conversational AIs lack the ability to generate content. Advanced AIs such as Chat GPT and Bard combine generative and conversational AI, making them much more attractive to the general public [19].

1.1. Generative AI in the Educational Process

With the popularization of AI, a lot of research has begun to emerge on how AI can enhance the learning process by improving and optimizing both face-to-face learning and blended and online instruction.

A study of the popularization of online courses and blended learning during the COVID-19 pandemic [10] found that while these types of learning are significant, there are a number of challenges that AI can help solve. For example, an individualized approach to students. During online learning, teachers cannot always track how effective the information is. This problem can be solved by using AI to "tailor" information to the student's needs. Moreover, personalized learning can be facilitated by AI-powered tools that identify individual learning gaps and recommend content tailored to each student's unique context. AI-powered tools can also assist instructors in generating assessments and issuing grades and feedback automatically. This allows teachers to spend more time on actual teaching and student interaction. In addition, AI can help revolutionize assessments by evaluating students'

performance over a more extended period, ultimately providing a fairer and more comprehensive evaluation system [11].

Intelligent learning systems, another area where AI is making strides, can mimic teachers, providing students with a personalized learning experience. Although they may be limited in their coverage of a particular field, these systems have proven effective in improving student performance and providing students with the right content.

However, the success of AI depends heavily on the availability and quality of student data, which has historically been a challenge for education companies as well as data researchers. The COVID-19 pandemic has dramatically expanded the use of educational products, thereby providing more data for AI systems. This data, in turn, has the potential to improve further and refine AI-based learning tools and methods.

In addition to the benefits already mentioned, AI technologies offer educators new opportunities to improve teaching methods, including learning analytics. Teachers can increase their effectiveness by using AI tools, and they can also promote self-regulation among their students. Despite these benefits, many educators have not yet used AI-driven technologies. Thus, it is critical to understand what the transition to using such technologies will entail and what obstacles they may face in their implementation. We must provide them with the digital competencies needed to enrich students' learning experiences.

Teachers' concerns about the introduction of AI into the classroom should also be taken into account. Some fear that they will be replaced, believing that AI will weaken their professional positions [12]. This may contribute to reluctance and slow down the adoption of AI technologies, as teachers may not understand that AI can complement, not replace them.

Teachers may hesitate to adopt AI-based tools because they often lack a clear understanding of how student data privacy is ensured and how to avoid algorithmic bias. In addition, a lack of infrastructure, funding, and support is often an obstacle to the widespread adoption of AI systems. Finally, teachers may not have the necessary technical and pedagogical knowledge and experience to successfully integrate AI tools into the classroom, which may cause them to be hesitant and resistant to using them [19].

1.2. Effective use of Prompt Engineering in the Educational Context

Prompt engineering is the process of structuring text that can be interpreted and understood by a generative AI mode [16], [17].

It is crucial to understand that the effectiveness of AI language models is influenced not only by the algorithms they use and the training data, but also by the quality of the instructions they receive [13], [15].

The capabilities of generative AI can be dramatically expanded with careful prompt engineering. Teachers can improve the assessment process, students can receive accurate, contextually relevant information. However, it should be realized that in some cases, users can even extract results from the generative AI model that it is not allowed to generate. This is

known as reverse engineering or jailbreaking [14]. It is very important to take these features and concerns into account when integrating AI into the learning process.

Generative AI can contribute to the creation of learning content. However, communication and interaction between humans and generative AI revolve around engaging in critical conversations between these entities, which emphasizes the importance of creating appropriate prompts that require us to understand prompt engineering.

The goal of prompt engineering is to improve the model's responses by adapting them to the structure, content, and tone of the question. This, in turn, helps to achieve more accurate, valuable, or consistent answers. In the field of AI language models, prompt development involves creating well-designed prompts that elicit predictable responses, ensuring clarity, relevance, and accuracy.

To write good promos, the following strategies can be considered:

1. Clarify the objective: Explicitly state the intended purpose of your prompt. What type of response or information do you aim to obtain? Clearly define the desired learning outcome or the interaction you seek.

2. Be direct and concise: Create prompts that are clear, concise, and free of ambiguity.

3. Provide context: Establish the setting or background for your prompt, allowing the AI model to better grasp the task or subject matter. Contextual cues aid in guiding the model's response and maintaining relevance.

4. Offer examples: If possible, furnish illustrations of the output you expect from the language model.

5. Specify the format: If you require a particular response format or structure, make sure to explicitly state it in the prompt. For instance, if you need a step-by-step answer or a pros-and-cons analysis, provide clear instructions accordingly.

6. Refine, optimize, and debug prompts: Continuously fine-tune your prompts to elicit more accurate, relevant, and contextually suitable responses from the model [18].

By adhering to these tactics, educators, researchers, and users can enhance the art of prompt engineering, yielding meaningful and precise responses from AI language models while aligning with their unique goals and prerequisites. From this perspective, it can be argued that in education, prompt engineering can serve as a tool to stimulate critical thinking, and nurture a deeper understanding of the subject matter.

Generative AI full power becomes visible when it is guided by human prompts. Under human guidance, generative AI opens up endless creative possibilities. The key to ensuring effective communication and interaction between humans and generative AI is the skillful creation of prompts. The skillful creation and engineering of appropriate prompts is of paramount importance, as they directly affect the capabilities of generative AI [19].

1.3. Chapter 1 Summary

Generative AI has significant potential to revolutionize the educational process by offering personalized learning experiences, improving assessments, and optimizing teaching methods. The role of prompt engineering is crucial to ensure effective communication between users and AI systems.

However, the integration of generative AI into the educational process also comes with certain challenges and risks, such as data privacy, security, potential misinformation, and the impact on human interaction. Addressing these issues and equipping teachers with the necessary digital competencies are important steps to fully utilize the potential of generative AI and rapid engineering, which will ultimately contribute to enriching and engaging students in the learning process.

Nevertheless, it is important to understand that the exploration of Prompt engineering and engagement with Generative Models are in preliminary stages and could undergo significant changes over time [19].

To summarize the main problem, it boils down to the question: "What is the most efficient and convenient way to use a large language model to create learning materials and training courses?"

In answer to this question, this research proposes the development of an Interoperable web-based system that leverages the capabilities of the Chat GPT API and Prompt Engineering to create learning resources.

In addition, this research aims to consider potential areas where the proposed system can be optimally utilized.

Chapter 2. Capstone Project: AI-based Interoperable System for Generation of Learning Materials

2.1. Solution Overview

The education sector requires innovative solutions to accelerate the learning process without compromising its effectiveness. The proposed solution, 'Interoperable Web System for

Generation of Learning Resources on the Base of LLM and Prompt Engineering,' aims to create a new system based on Generative AI that will help make the process of creating educational materials more efficient.

Creating and updating learning materials is a time-consuming and resource-intensive task. This system aims to enhance the process by utilizing Chat GPT's capabilities, as it is currently the most advanced Generative AI.

2.2. Requirements for AI-based Learning Materials Generation System

2.2.1. Business Value of Web System for Generation of Learning Materials

The creation of training materials is a labor-intensive procedure that requires a significant amount of human resources and time. The project aims to automate this procedure using artificial intelligence, allowing human resources to focus on content validation and reducing the time required for material creation.

Business Drivers:

- Optimization of the process of creation and update of learning materials.
- Reduce the time required to prepare new content.

Business Goals:

- Increasing the number of learning materials and their variability.
- Reducing the workload of instructors.
- Creation of a portable application that can be easily integrated with existing systems.

Business Objectives:

- Develop an LLM-powered application that generates educational content based on user instruction and exports the results in JSON format.

2.2.2. Architecturally Significant Requirements of the System

ASR		Rationale	Type	Priority
1	Automatic generation of learning materials	To increase efficiency by reducing the time spent and human resources needed for creating learning materials.	Functional	High

2	Increasing the number of learning materials and their variability	Generation a diverse range of learning materials to cater to various educational needs.	Functional	High
3	Reduce workload of instructors	The system should automate the process of creating learning materials, thereby reducing the workload of instructors.	Usability	Medium
4	Exports the results in predefined format	Export the generated content in a universally accepted format such as JSON for easy integration and data exchange.	Constraints	Medium
5	Standard Protocols Support	The application should support standard communication and data transfer protocols (like HTTP/HTTPS) to ensure interoperability.	Constraints	High
6	Scalability	The system should be able to meet the growing demand of users without compromising performance and degrading generation speed.	Quality attribute	High
7	Portability	The application should be portable and easy to integrate with existing systems to optimize deployment and usage.	Quality attribute	High

Table 1. List of ASRs.

The system operates under the following assumptions: materials are generated:

- Materials generation begins upon receiving a user request, including the necessary instructions.
- The primary models for content generation are GPT 3.5 and GPT 4.

2.2.3. Non-functional requirements

This section presents a structured method for identifying, evaluating, and documenting non-functional requirements. The aim is to guarantee that essential non-functional aspects are comprehensively understood and integrated into the architectural design.

Efficiency	*The system must automatically generate learning materials with minimal delay, optimizing the use of computing resources to complete tasks within*

	a reasonable time frame.
Usability	The system should provide an easy-to-use interface for teachers to simplify the creation of learning materials and reduce their workload. Additionally, the system should offer clear instructions and assistance to ensure users can complete tasks with ease.
Reliability	The system should demonstrate high reliability in producing accurate and relevant training materials, reducing the likelihood of system failures that could interfere with continuous service availability.

2.3. Quality Attributes

2.3.1. Descriptions of Quality Attributes

This section presents the key quality attributes for the system.

Quality attribute	Business POV	Architectural POV
Scalability	The system should handle growing user demand without impeding the generation process.	Scalability impacts architectural design, like the use of technologies that enable scalability and parallel request processing, potentially through the use of scalable cloud-based solutions and effective load-balancing techniques.
Portability	The application should be easily integrated with existing systems to simplify the deployment and usage process.	Architecturally, portability can be enhanced by utilizing platform-independent design and development strategies, such as using container-based solutions like Docker. This ensures ease of deployment across diverse systems and environments.

Table 2. Quality attributes from business and architectural Point of views.

2.4. Solution Architecture of the AI-based Learning Materials Generation Web System

This section contains a target architecture of the "AI-based Learning Materials Generation Web System" project, which describes how the system is built and interacts with external systems to achieve the project's requirements.

2.4.1. AI-Based Web System Technology Stack Justification

The proposed system is architecturally designed on the basis of the Python programming language. Python has a number of advantages that make it the best choice for this system. Python also has extensive support for Large Language Models and related technologies, such as Artificial Intelligence, Natural Language Processing, and Machine Learning, which increases its usefulness.

Application uses the open-source Postgres database for data storage needs. This relational database system provides significant speed and reliability, which is critical for the project purposes. In addition to reliability, Postgres offers many functionalities, such as complex queries, foreign keys, triggers, and views, which makes it easier to manage and interact with data. Postgres is ACID compliant [20] and meets our needs for reliable data storage.

The system web interface is built on JavaScript along with the React framework. JavaScript is the main language for user interface development and provides dynamic and interactive elements for web applications. Using React, which is built on JavaScript, adds more power, flexibility, and efficiency to our web interface. React's component-based architecture enables modular development and increased reusability, optimizing both development time and performance. Virtual DOM [22] capabilities highly improve performance by minimizing DOM manipulation.

Another important part of the system architecture is integration with the Open AI API. Serving as the basis for creating training materials and courses, the OpenAI API [21] is crucial to the system's operation. This API allows system to use the power of the advanced natural language processing models developed by Open AI, offering resources for machine learning algorithms and language models.

2.4.2. Identified Project Risks

The main system functionality - content generation relies on Open AI API and utilizes two versions of LLMs:

- GPT 3.5-turbo
- GPT 4-turbo

The use of this API is paid, and the cost depends on the intensity of use (the size of I/O tokens) and the LLM model.

Model	Input tokens	Output tokens

GPT 3.5-turbo	$0.01 / 1K tokens	$0.03 / 1K tokens
GPT 4-turbo	$0.0010 / 1K tokens	$0.0020 / 1K tokens

Table 3. GPT models pricing comparison

The GPT 4 model is significantly more expensive than the GPT 3, but it has the potential to show better generation results due to advanced generation algorithms and better training data.

One potential risk is that the system's generation capabilities depend wholly on the Open AI API. If the API is unavailable, the system's functionality may be severely limited.

2.5. High-Level Solution Architecture of the AI-based Learning Materials Generation Web System

2.5.1. Context Diagram

The Context diagram presents a high-level overview of the software system, demonstrating its interactions with end users and external systems.

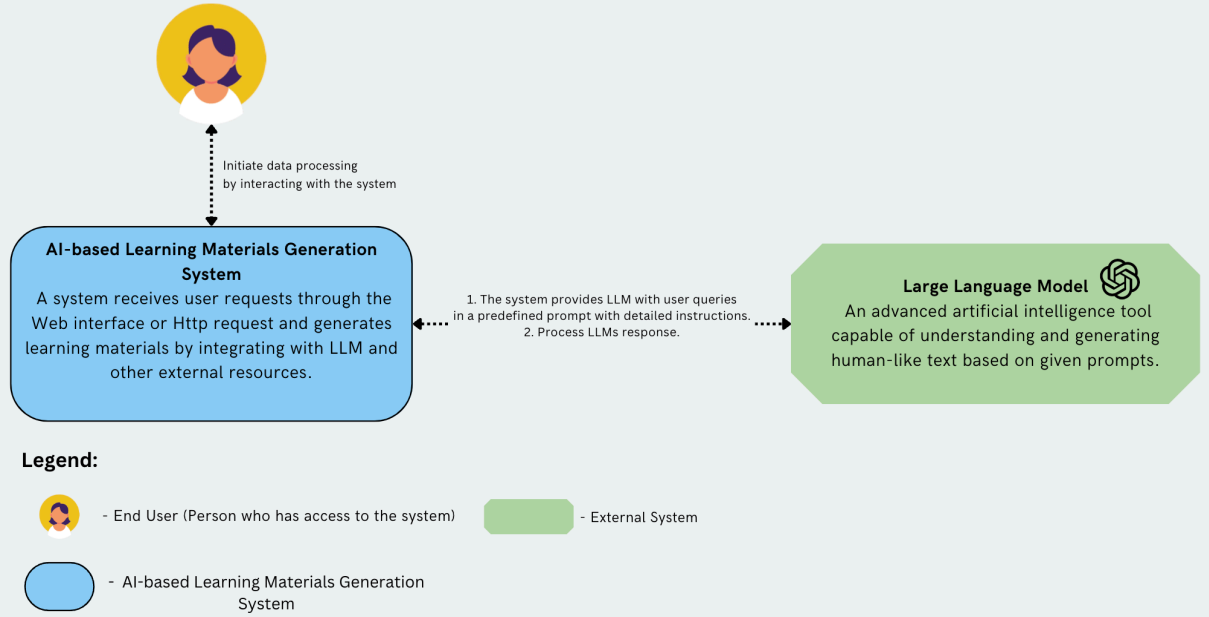

Figure 1. - Contextual diagram of the system

The AI-based Learning Materials Generation Web System application generates learning content based on user instructions, injecting them into a predefined prompt sent to the LLM via a REST call and then returns the result to the user.

2.5.2. Container Diagram

The diagram below illustrates the high-level structure of the architecture and the distribution of responsibilities.

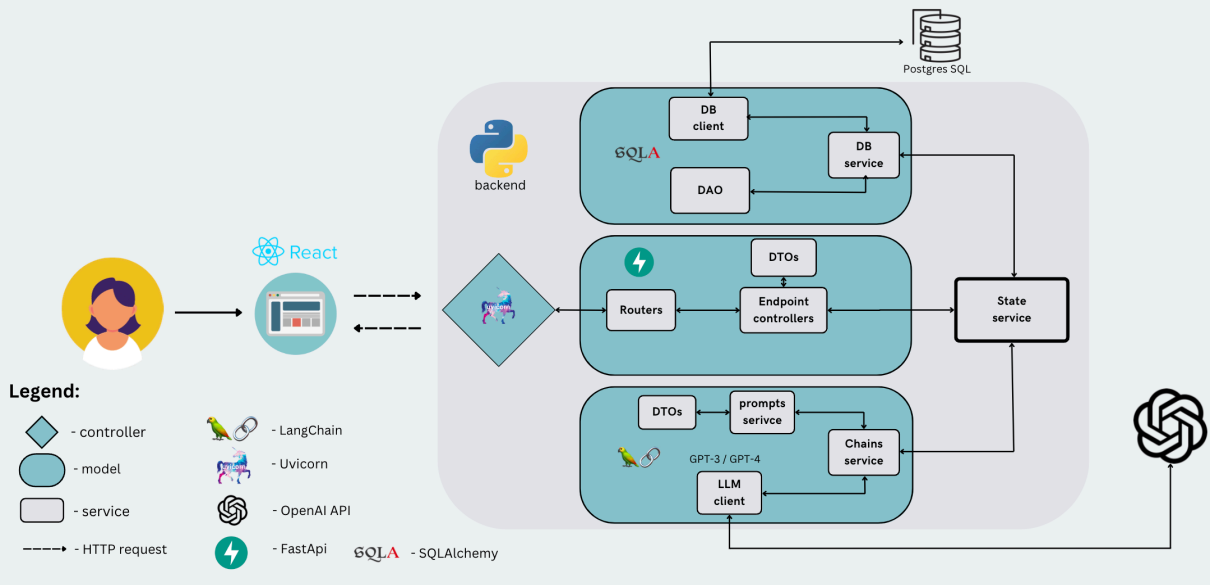

Figure 2. – Container diagram of the system

System interaction begins when a user forms a query on a topic of interest via the web interface. This user-initiated action triggers an HTTP request that redirected to a system backend. The server side handled by Uvicorn web server [23] receives the request, which acts as an orchestrator. Here, the user's request is routed to an HTTP controller. The router is responsible for deciding which controller should be responsible for initializing the next operation. When the initialization is complete the State Service oversees this execution process. In addition to overseeing the overall workflow, State Service also assigns specific subtasks to the Database and Large Language Model modules. Both modules play a key role in managing subprocesses in the system. The Database module contains the DataBase client communicating with the Postgres database and the Data Access Objects (DAO) layer.

The LLM module handles another sub-tasks triggered by State Service. It comprises three main components that are pivotal for managing AI-powered interactions. This includes a Chains Service, which orchestrates generation pipelines, an LLM cCient that interfaces with the Open AI API via HTTP requests, and a Prompt Service responsible for generating the final prompt using user data and instructions. Upon completion of the above tasks, the response received from the Open AI API, a State Service, triggers updates in the Database via the Database Module.

2.6. Implementation Details of the AI-based Learning Materials Generation Web System

2.6.1. AI-based Learning Materials Generation System Web Interface

The web interface of the application is based on JavaScript and the React framework.

React, a Facebook product, has several important features that make it an exceptional tool for front-end development.

- Modular Architecture: React uses a component-based architecture, making application development more flexible and efficient. Components are reusable and can manage their state.
- Optimized rendering: React uses a virtual DOM that effectively improves application performance by optimizing the rendering process. It distinguishes between the existing DOM and the new DOM, updating only the differences rather than refreshing the entire page, making the update process faster and more efficient.
- JavaScript Library: React is a JavaScript library, it gives the flexibility to choose the libraries, tools, and architecture for developing an application according to their requirements.
- Acceleration and performance: React allows to use of separate segments of applications on both the client and server sides. This dual use significantly speeds up the development process and increases efficiency.

In addition to React, the web interface also uses TypeScript to improve code clarity and quality. TypeScript's static typing feature provides a robust code structure that improves readability and efficient handling of data types. This results in code that is more maintainable and less contain less errors, which contributes to the overall reliability of the application.

The web interface of the application has several key features, including navigating through existing materials, creating new materials, presenting and downloading.

Learning Material Search Capabilities via Web Interface. Users can perform a search by entering the name of a course in the input field and initiate the search by clicking the "Submit" button. This action triggers a function that sends a request to the endpoint of the application backend.

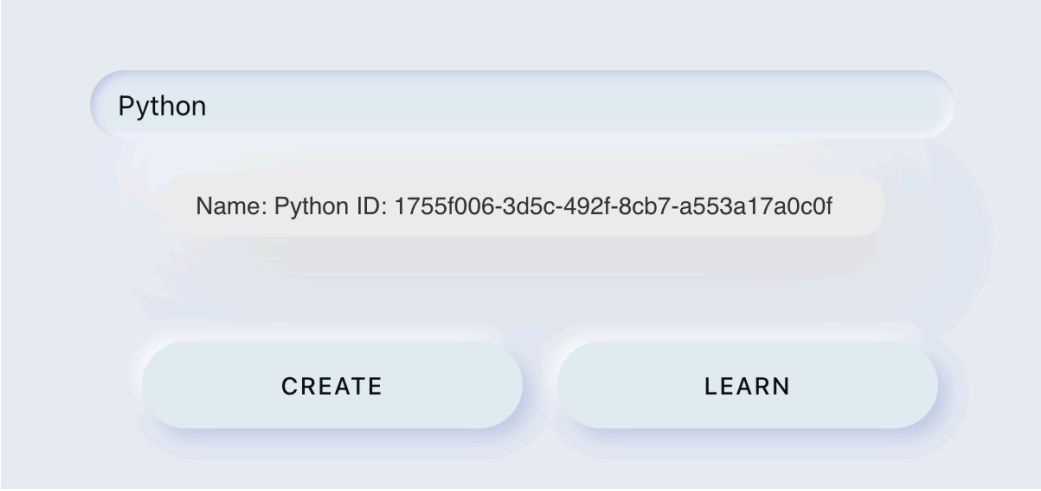

Figure 3. – The result of a search query by the name of the course "Python".

The response from the server side can be processed in several scenarios:

1. In the success scenario where the backend returns a status code 200 and a response in JSON format, the web interface processes the response and shows a drop-down list with a parsed list of relevant courses. Each element of the list has next format: Name:

${course_name}, ID: ${course_uuid}, where "Name" corresponds to the full name of the course and "ID" stands for a A Universally Unique Identifier (UUID) that is assigned to each course during the generation process. This identifier helps distinguish between courses that may have the same name. When a user selects a course from the list, another request to retrieve the entire course is performed. After the response is returned, the Course Tab with data becomes visible. When a user selects a course from the list, another request to retrieve the entire course is performed, displayed on the Course tab that appears afterward.

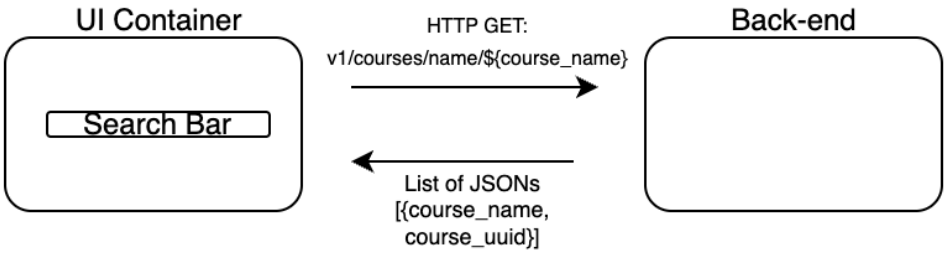

Figure 4. – Search by course name flow

2. The back-end returns status code 404. The web interface shows user a notification window with the message "Course not found", in case of an error and status code 500, a window with the notification "There was an error!" will appear.

Learning Material Generation Capabilities via Web Interface. Another feature of the application is material generation (course creation). It offers two main user workflows along with several additional options.

1. In the first case, user can specify only the course name ${course_name} he would like to generate.

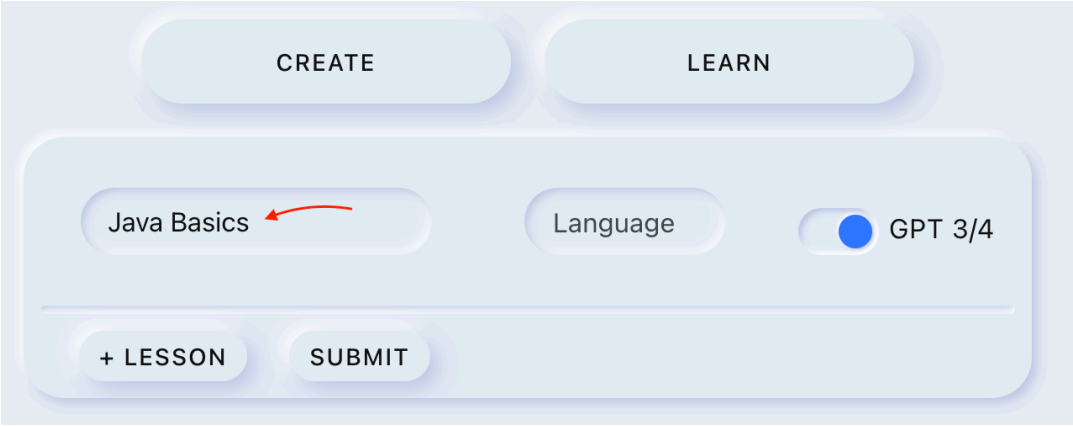

Figure 5. – Material generation by course name only

2. The second workflow allows users to provide the name of the course ${course_name} and additional information. This could be a list of lessons, represented as [${lesson_1}, ${lesson_2}]. Each lesson should have the required field "lesson title" ${title} and an optional field list of topics [${topic_1}, ${topic_1}]. In this case,

instead of a course with general information, a course will be created only for specific lessons with specific topics related to the specified course ${course_name}.

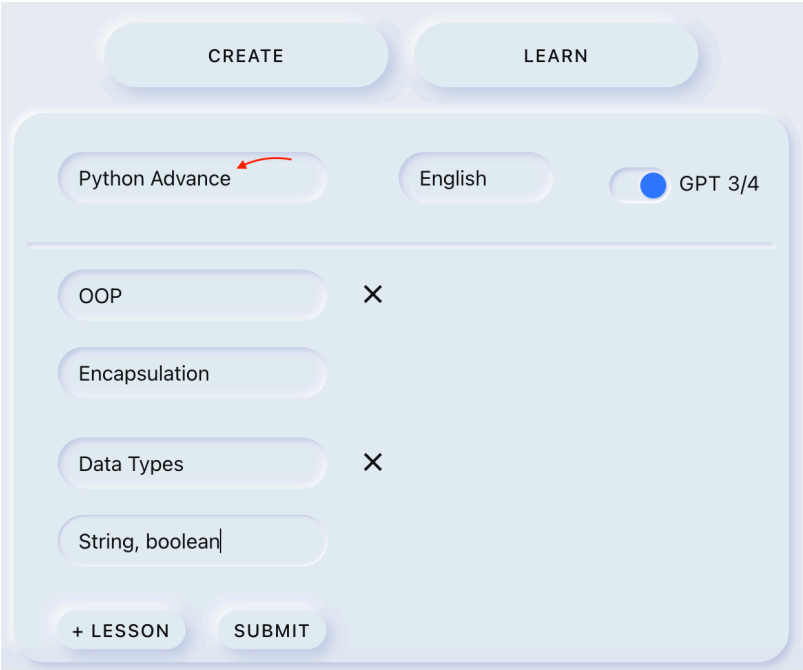

Figure 6. – Generation by course name and additional information

After a user has finished forming the request and submit it, a POST request with body in JSON format is sent to the backend to initiate the generation of training materials. In response, the user receives a UUID that can be used to identify the newly generated course since the generation of materials can take up to 5 minutes.

The system will not keep the connection between the front-end and back-end. It will immediately return the generated UUID to the user, which a user can use to access course materials after the generation is complete.

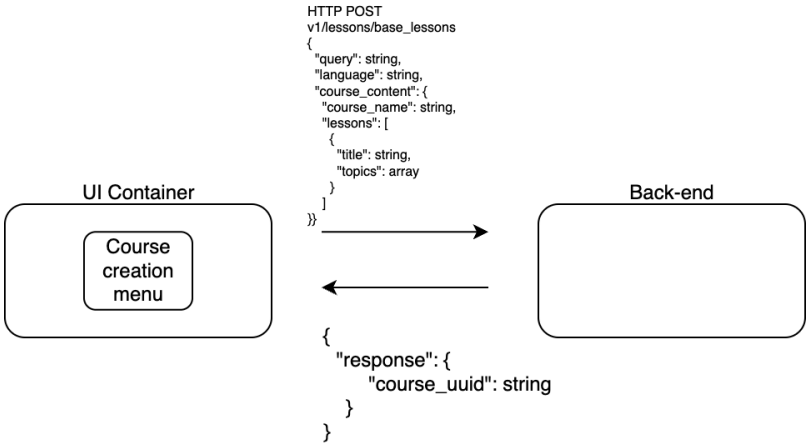

Figure 7. – Generate new course flow

Additional features provided to the user include the ability to select the language for course generation and the ability to choose between GPT versions 3 or 4. It is important to note that the choice of these two options can have a significant impact on the quality and cost of the created materials. For example, GPT 4, although it has the advantage of providing more detailed answers, is more expensive. The choice of generation language also affects the price.

Retrieval of Learning Material by Unique Identifier via Web Interface. The third important function receiving the generated data by a UUID.

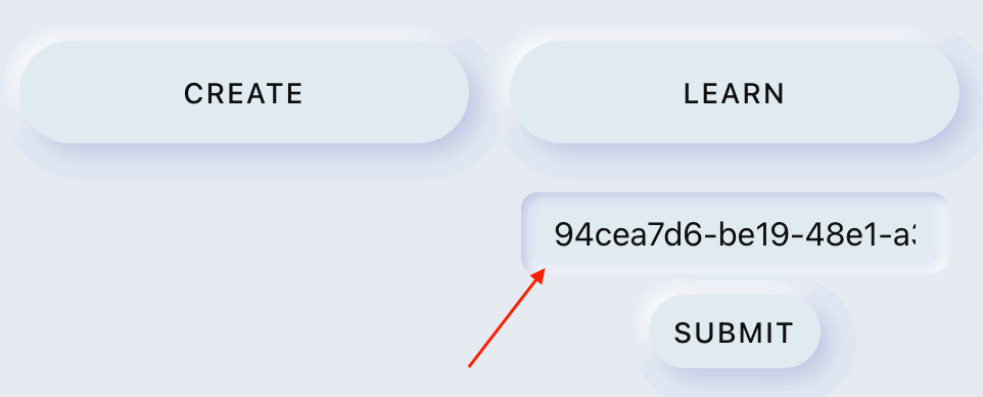

Figure 8. – Receiving generated materials by UUID

Using this function, a user can get information on the status of the generation of a particular course, as well as get an already generated course.

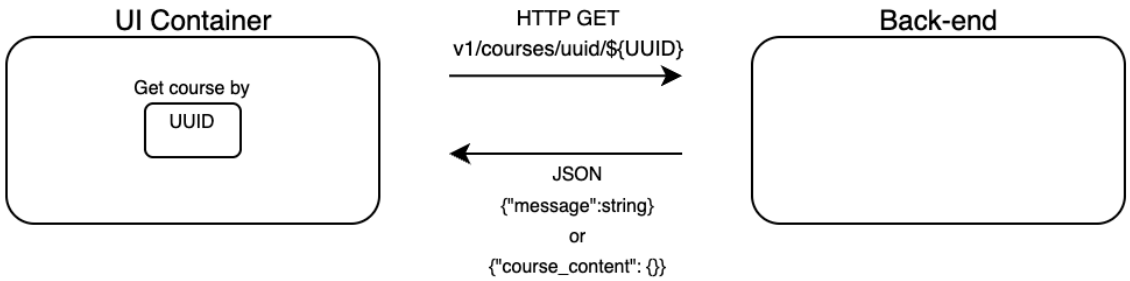

Figure 9. – Get course by UUID flow

There are three possible response statuses:

1. Processing - generation of materials is still in progress
2. Generated - the generation of materials is completed, the user receives the materials
3. Failed - the generation of materials was not successful, you need to repeat the request

Learning Material Representation. The pivotal segment of the web interface is the Course Tab, dedicated to displaying the generated course materials.

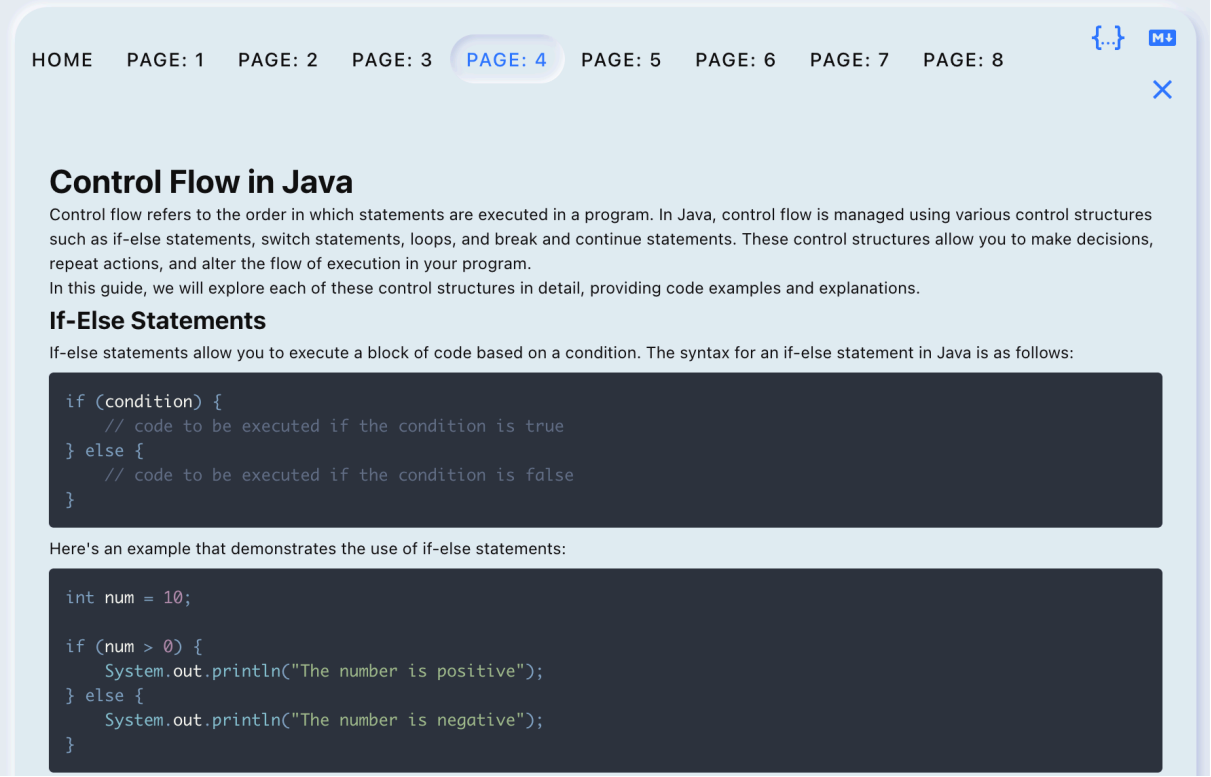

Figure 10. – Get course by UUID flow

Initially, user see a page where he can access the entire course content, organized on a one-page-per-lesson basis, including all its Topics.

User can study the materials directly from the web interface or download them in either Markdown or JSON format for offline use. The JSON format, being a universal format, allows for future reuse, integration with other services, or custom user implementation.

As alternative, the Markdown format was selected. This format allows user to utilize the materials without additional processing or formatting. Notably, Markdown is widely supported by numerous text editors like IntelliJ, Sublime Text, VS Code, and apps like Notion.

2.6.2. Learning Material Generation Requests Processing

The HTTP controller acts as the entry point to the application's backend, serving as the primary interface for all interactions via HTTP requests and responses between the client and server sides. It is designed to route incoming requests to the appropriate processes and return responses.

This module consists of two key components: Pydantic models and Controllers. Pydantic models help structure and validate incoming data, while Controllers, in particular Course Controller and Lessons Controller, handle the functionalities related to courses and lessons, including generation of materials and retrieving course or lesson data.

Pydantic models are essentially data classes that validate input data. Pydantic provides runtime type checking and helpful error messages when data is incorrect.

Models:

1. The Lesson model defines a "Lesson" object. It consists of a "title" field of type string, which indicates the lesson's name. It also has a "topics" field, a list of string types containing all the topics related to the lesson.

2. The CourseContent model defines a "CourseContent" object. It contains two fields. The "course_name" field is a string data type containing the course's name. It also has a "lessons" field, which is a list of "Lesson" objects defined in the "Lesson" model, indicating that the course contains several lessons. This characteristic of a nested model allows handling of complex data types in a structured way.

3. The Course model represents a Course object. It includes a "course_content" field containing a CourseContent object that encapsulates all the content relevant to the course. It also has a "language" field of type Optional[string], meaning it can be either a string or None. The "language" field also has a default value set to "English," this language will be used for course generation if a user does not put any value on his side.

Mapping data models to classes is a common practice. This allows to display data structures in OOP style. Such approach makes possible to reuse data models, which makes the code more readable and structured. In addition, it allows you to make changes and process new data structures faster.

Controllers perform request routing, which defines what logic will be applied to the user's request. At its core, it uses FastAPI's "APIRouter" to set up routes.

Course Controller supports several endpoints end has next set of functions.

1. *Generate_response* function. An asynchronous private function that starts a workflow to process a course request and adds it to the background tasks. It generates a unique UUID for the course, logs the incoming request, and triggers the *process_course_request* in the background. Finally, it returns a response with the generated course UUID.

2. *Process_course_request* function. This function calls the *Process_course_request* function in StateService to start data generation based on the provided inputs.


```
{
   "query":"Java basics",
   "language": "English"
}
```


Figure 11. – /courses/base_course request example

3. *Generate_base_course* and *Generate_advance_course* functions. These are asynchronous route handler functions linked to POST routes "/courses/base_course" and "/courses/advance_course" respectively. It calls the *Generate_response* function using the data from the request and adds it to the background tasks. The difference lies in the ${llm_version} value. The former uses GPT v.3, and the latter uses GPT v.4.

4. *Get_course_by_uuid and Get_course_by_name* function: These are route handlers for GET requests aimed at fetching course information by either UUID or Course name. If no course data is found, an HTTPException with status code 404 is raised, indicating "Course not found".

Lesson Controller has similar functionality to the Course Controller. It processes requests that have a more complex structure and include additional objects from Lessons and Topics.

1. *Generate_response* function: An asynchronous function designed to process the lesson-related requests. It first assigns a unique UUID to the course, logs the incoming request, initiates asynchronous execution of *Process_lessons_request* function by adding it to the background tasks, and finally returns an HTTP response carrying the UUID.

2. *Process_lessons_request* function: This function calls the *Process_lessons_request* function in StateService with the received request's details.

```
{
   "query": "python-tutorial",
   "language": "English",
   "course_content": {
      "course_name": "Python Tutorial",
      "lessons": [
         {
            "title": "Data types",
            "topics": ["String", "boolean"]
         },
         {
            "title": "OOP",
            "topics": ["Incapsulation", "Inheritance"]
         }
      ]
   }
}
```

Figure 12. – /lessons/base_lessons request example

3. *Generate_base_lessons* and *Generate_advance_lessons* functions: These are asynchronous functions tied to POST routes "/lessons/base_lessons" and "/lessons/advance_lessons" respectively. They call *Generate_response* function with the data from the incoming request and the relevant ${llm_version} (either GPT v.3 or GPT v.4), then add it to the background tasks.

4. *Get_course_by_uuid* and *Get_lessons_by_uuid* functions: Those function handles GET requests aiming to obtain lesson details based on UUID. If no matching lesson data is found,

an HTTPException with a 404 status code is raised, indicating that the relevant course has not been found.

2.6.3. Trigger and Monitoring of the Materials Generation Process Within The System

StateService is associated with the management and control of key processes related to request processing and course creation, as well as interaction with other elements of the system. It uses the Singleton pattern to provide a single, shared instance of the StateService throughout the application.

Upon initialization, StateService creates instances of CourseChain, LessonChain, and DbController. CourseChain and LessonChain manage the generation of course and lesson materials, and DbController handles database interactions.

In the *Process_course_request* function, the StateService calls DbController to initialize, creating a new row in the database with request data and related information. At this stage, the course is assigned the "Processing" status. The course will be in this status until the LLM Module updates it after completing its part.

Then StateService calls CourseChain to start generating course materials and corresponding records and initiates the generation of lesson content. In this stage status still is "Processing".

Finally, when CourseChain finishes, StateService updates the course generation status to "Generated" and saves the course data.

The *Process_lessons_request* function works in a similar way: it writes the request data to the database, starts generating materials for lessons, updates the course generation status to "Created", and saves the lesson data.

Both functions have error handling mechanisms that log errors and update the course generation status to "Failed" in case of exceptions.

Finally, the *Get_course_by_uuid*, *Get_course_by_name*, and *Get_lessons_by_uuid* functions are used to get course or lesson data based on a UUID or course name.

2.6.4. Storing and Accessing Learning Materials Through Database Interaction

A DataBase Controller consists of several components that handle the connection to a database, store information, and structure it.

The DbClient under the DataBase Controller is the primary interface for interacting with the application's database. It leverages SQLAlchemy's Engine to handle the connection pool and SQL dialect. The client utilizes the Singleton pattern to ensure a single instance is used across the application. Once initialized, DbClient creates a connection with the database and initiates an ORM session. Several functions are utilized for database interaction. For instance, *Create_tables* facilitates table creation as per the metadata, Get_session returns the current

database session, *Add_to_db* adds new objects to the database, and *Get_from_db* fetches database information based on specified filters.

The next component is the database tables. They control the structural organization of the data in the application database. The database has four main tables: StateDb, CourseDb, LessonDb, and LessonContentDb. StateDb is a state management table that stores key elements such as query text, course UUID, LLM version, language, and generation status.

CourseDb is central to course-related information, with fields such as “course_name“ and “course_uuid”. It also establishes a relationship with the StateDb table.

LessonDb focuses on lesson data and contains fields such as “'esson_name”, “topics”, and “course_id”. It also connects to the CourseDb and has a relationship with the LessonContentDb.

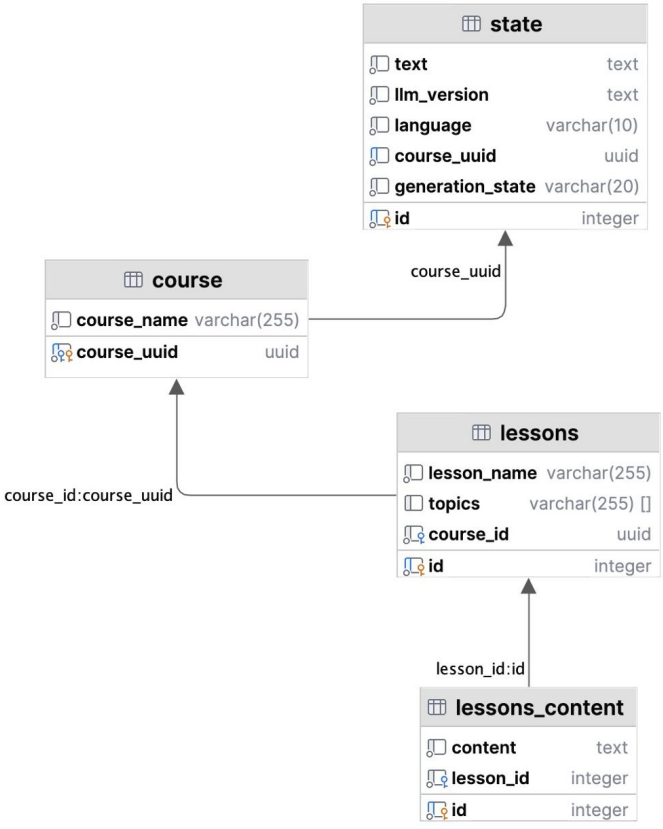

Figure 13. – Database tables schema

Finally, the LessonContentDb is used to store lesson content. It maintains a relationship with the LessonDb table to match lesson content with the corresponding lessons.

The DbController serves as the main communication channel between the application and the database, managing changes and data retrieval. By using the DbClient to interact with the database, the class helps update the course creation status and write status data, course records, and lesson content.

DbController controls the course update by tracking the states of the new generation. It defines a StateDb object with query parameters to record state data and adds it to the database.

It also manages course records by defining CourseDb and LessonDb objects according to the course response and writing them to the database. In addition to writing operations, the DbController retrieves data by course UUID or course name, checks the course creation status, and matches lessons and their content to the correct course IDs. It also handles exceptions and returns appropriate error messages when necessary.

2.6.5. Integration With External LLM Services, Formation and Handling of the Generation Request

LLM controller includes three large submodules: Prompts, LLM Connector, Chains. Each of them is directly involved in preparing the request for materials and processing the responses.

Integration With Open AI API. The LLM Connector is designed to connect to the OpenAI API. This module consists of two components: "OpenAI4Model" and "OpenAI3Model". Both components extend the "ChatOpenAI" class from the LangChain framework, which is responsible for communicating with the API. These components define parameters such as request timeout, temperature, retries, and model type. OpenAI4Model and OpenAI3Model have the same structure and purpose, differing only in the name of the model they use from the program configuration. These classes, which are run by the LLM controller, manage the interaction with the corresponding models of the Open AI API.

Formation of the Final Prompt for LLM Based on Predefined Instructions. The Prompts submodule forms another critical segment of the LLM Controller. It contains templates of prompts designed for generating courses (course_prompt) and lessons (lessons_prompt).

```
""" ### Instructions ###
If MUST ONLY generate a step-by-step self-study course based on the above topics".
You MUST ALWAYS answer in literature {language} language even if the user questions in another language.
You MUST give very precise instructions.
You MUST include Topic-related terms in instructions.
You MUST be specific, do not give general advice.
You MUST follow the RULES within the section if provided.
### Instructions ###
### Format ###
{format_instructions}
### Format ###
Topics: '"{query}"' """
```

Figure 14. – Course generation prompt

These prompts include instructions to reduce the risk of model hallucinations by clearly formulating the task, for example, "You MUST provide detailed code examples with explanations." "You MUST be specific, do not give general advice."

Also, the prompt indicates the format in which the LLM should return the answer. The response format for all types of materials is defined as JSON. Format structure is built with the help of Pydentic classes, similar to the approach mentioned earlier in this paper.

Defining and Initializing Learning Materials Generation Stages. The Chains submodule in the LLM Controller defines generation steps. CourseChain is responsible for generating courses without the use of nested Lessons that are specified by the user.

This class call the creation of the final Prompt and provide it with user data, determines which Large Language Model will be used to process the request and launch generation. Upon response from the model, CourseChain processes the text by parsing JSON from it, and returns a dictionary with the generated course content.

The LessonsChain class in the Chains submodule of the LLM controller is responsible for the orchestration of lessons creation using OpenAI models. Like CourseChain, LessonsChain inherits the base Chain class.

Unlike CourseChain, LessonsChain performs bulk lesson creation using parallel request processing. Efficiency is achieved by simultaneously preparing and sending multiple lesson creation requests and merging the lesson content.

In the *call* function, LessonsChain converts the course content into individual lessons. Then, it generates a list of prompts, so each lesson has a separate prompt that will be used in the feature.

All prepared lesson prompts are called using the *RunnableParallel.invoke* function from LangChain framework. This function generates a separate request for each lesson and performs them in parallel. Parallel processing makes the generation of lessons faster and more efficient.

Experiments show that generating an extensive course with more than ten lessons can take nearly half an hour without Parallel execution. At the same time, using Parallel execution, this process takes about five minutes.

After receiving the answers, the class processes the results and returns a dictionary with the prepared course and lesson content.

Chapter 3. Results and Analysis

The code base of the described system can be found at the following links:

- Github-repository of the backend part of the described system: https://github.com/KakunArtem/svitoch
- Github-repository of the frontend part of the described system: https://github.com/KakunArtem/svitoch-ui

3.1. Conclusions About the Web Interface and Backend Architecture

It is important to note that clear presentation is critical for effective perception of learning materials. The system's web interface employs the 'Neumorphism' UI style, which is common and intuitive.

However, it currently only serves a demonstrative function. The software comprises fundamental functions and needs further development to provide a complete introduction to users.

The server-side architecture. The server-side architecture has demonstrated reliability and effective interaction with Open AI models to produce personalized learning materials and lessons. Asynchronous request execution enables the system to process multiple user requests simultaneously without slowing down.

The system successfully handles loads, ensuring consistent responses that do not overlap and recording information in the database without loss. Error handling and sophisticated logging are essential for preventing system crashes and tracking events and user requests. The server side utilizes the REST [24] software architectural style, which facilitates integration with other systems and services. This enhances the system's flexibility and makes it more readily integrable out of the box.

3.2. Conclusions About the Quality of Generated Materials

The system's key feature is the generation of training materials, which is supported by all processes and components, from the web user interface to database tables. It is important to note that the quality of the generated materials varies depending on the GPT model used for generation. The differences are mainly in the details of the courses and code examples. GPT 4 offers more detailed explanations and examples compared to GPT 3. However, it is important to note that the quality of generated material improves with more detailed specifications of lessons and topics.

The speed of generation is also affected by the level of detail provided. The cost of a request increases with the number of materials expected. At the time of writing, the application uses GPT 4, which was released several months ago and is quite expensive [25].

Generation of the course of 20 + pages show the following cost results:

Course name	Model Type	Cost $
C# Advance	GPT 4	$4.14
C# Basic	GPT 3	0.27
CSS Advanced	GPT 4	$5.00
CSS Basic	GPT 3	$0.35

Table 4. Course generation costs

It is important to note that while the cost of GPT 3 is lower, the quality of the generated materials is also lower. In the scope of this work, a comparison of generation price to quality of materials was not performed.

Prompt engineering remains at the state-of-the-art stage. This relatively new area is under active development and has no clear postulates. In addition, prompt optimization techniques can vary from one generative model to another. This work used several techniques to optimize integration with the Open AI model and reduce hallucinations.

- Dividing instructions into sections
- Clearly written instructions
- Formatting of responses

These techniques helped to achieve the desired outcome and good generation results. Although at this stage of development, the system can generate material for self-study. But it still needs moderation to be used in the most efficient way. Working with a subject matter expert who could edit or combine the final course with other materials would be the most effective solution not only for this system but for generation models in general.

3.3. Possible Improvement Points

At this stage, the system requires the implementation of a user authorization feature and a feature for tracking the authorship of educational materials.

Additionally, the system could benefit from the ability to update Prompts to modify its behavior through endpoints or admin panels without causing code change and redeployment of the whole system.

An additional possible feature is an analysis of professional skills. This can be a handy feature for professionals who want to have skills relevant to the job market. Analyze job vacancies in a particular field and identify the necessary skills to fill them, perform a cluster analysis based on the data obtained, and create materials relevant to that professional field.

Another potential feature is the inclusion of interactive communication with a chatbot trained to create a competency-based learning environment within a specific specialization. This would provide additional knowledge necessary to work effectively in the industry, as well as related knowledge that would be beneficial.

3.4. Results of the work done

During the project, we analyzed materials and publications on the current state of the art of using generative artificial intelligence in the educational process. In addition, highlighted best practices and the importance of using prompt engineering and related methods to achieve high-quality material generation. Covered risks expressed by researchers in publications and related materials. Implemented the AI-powered application for generation of learning materials.

Outcomes of the master's capstone project:

1. Proven the effectiveness of using Generative AI, particularly Open AI models, to generate educational materials and courses.

2. Implemented the application that uses LLM for text generation. For the project purpose, the usage of the GPT model was justified. The application interacts with LLM via OpenAI API and Postgres DB as data storage. Provided detailed architecture and demonstrated system context, container, and component alongside an overview of essential modules, libraries, and code examples. Differences in the usage of different models are shown in the example of the GPT-3 and GPT-4 models.

3. Described in detail the implementation of core features of the capstone project as prompt engineering, user interaction, and educational material generation. Mentioned that to improve the quality of the LLM output, the prompt should have clear and understandable instructions without ambiguity.

4. The rationale for using different dependencies, namely LangChain, React, and OpenAI API. Shown how the LangChain simplifies the interaction with LLM and provides an additional abstraction layer to work with any providers.

5. Shown how the implemented work can be extended and improved by adding additional features such as analysis of professional skills and interactive communication with a chatbot trained to create a competency-based learning environment.

References

1. A. Tong, Routers (2023) “Exclusive: ChatGPT traffic slips again for third month in a row”. Retrieved from: https://www.reuters.com/technology/chatgpt-traffic-slips-again-third-month-row-2023-09-07.
2. Harshita Chourasia (2023). Deep Learning. Publisher: Rubicon Publications, location_on4/4A Bloomsbury Square, Bloomsbury Square, London, WC1A 2RP, England ISBN: 978-1-80433-952-7
3. Sahoo S, Kumar S, Abedin M, Lim WM, Jakhar SK (2022). Deep Learning Applications in Manufacturing Operations: A Review of Trends and Ways Forward. Journal of Enterprise Information Management ISSN: 1741-0398. DOI: 10.1108/JEIM-01-2022-0025.
4. Weng Marc Lim, Asanka Gunasekara, Jessica Leigh Pallant, Jason Ian Pallant, Ekaterina Pechenkina. (2023). Generative AI and the future of education: Ragnarök or reformation? A paradoxical perspective from management educators. The International Journal of Management Education Volume 21. DOI: 10.1016/j.ijme.2023.100790.
5. Weng Marc Lim, Satish Kumar, Sanjeev Verma, Rijul Chaturvedi (2022). Alexa, what do we know about conversational commerce? Insights from a systematic literature review March 2022 Psychology and Marketing 39. DOI: 10.1002/mar.21654.
6. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio (2014). Generative Adversarial Networks. Advances in Neural Information Processing Systems. DOI: 10.1145/3422622.
7. Diederik P Kingma, Max Welling (2014). Auto-Encoding Variational Bayes. Conference: ICLR.
8. Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova (October 2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.
9. A. Radford, K.Narasimhan, T. Salimans, I. Sutskever (2018). Improving language understanding by generative pre-training.
10. Liu Kexin, Qu Yi, Song Xiaoou, Li Yan (202). Future Education Trend Learned From the Covid-19 Pandemic: Take ≪Artificial Intelligence≫ Online Course As an Example. Conference: 2020 International Conference on Artificial Intelligence and Education (ICAIE). DOI: 10.1109/ICAIE50891.2020.00032.
11. Olga Tapalova, Nadezhda Zhiyenbayeva, Dmitry Gura (2022). Artificial Intelligence in Education: AIEd for Personalised Learning Pathways December. The Electronic Journal of e-Learning 20(5):639-653. DOI: 10.34190/ejel.20.5.2597.
12. Wayne Holmes, Ilkka Tuomi (2022). State of the art and practice in AI in education. European Journal of Education 57(3). DOI: 10.1111/ejed.12533.
13. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G (2021). Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing.
14. Liu, Y., Deng, G., Xu, Z., Li, Y., Zheng, Y., Zhang, Y., Zhao, L., Zhang, T., & Liu, Y. (2023). Jailbreaking chatGPT via prompt engineering: An empirical study. DOI: 10.48550/arXiv.2305.13860.

15. Leo S. Lo (2023). The CLEAR path: A framework for enhancing information literacy through prompt engineering. DOI: 10.1016/j.acalib.2023.102720.
16. Diab M., Herrera J., Chernow B (2022). Stable Diffusion Prompt Book.
17. Albert Ziegler, John Berryman (2023). A developer's guide to prompt engineering and LLMs.
18. Aras Bozkurt, Ramesh C Sharma (2023). Generative AI and Prompt Engineering: The Art of Whispering to Let the Genie Out of the Algorithmic World. DOI: 10.5281/zenodo.8174941.
19. Kakun A., Tytenko S., "Generative ai and prompt engineering in education" Modern engineering and innovative technologies 29-03 (2023): 117-121. DOI: 10.30890/2567-5273.2023-29-01-052.
20. Wikipedia “ACID”. Retrieved from https://en.wikipedia.org/wiki/Acid.
21. Open AI Platform Overview. Retrieved from https://platform.openai.com/docs/overview.
22. Developer Mozilla “Document Object Model”. Retrieved from https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model.
23. Uvicorn “Inroduction”. Retrieved from https://www.uvicorn.org/.
24. Wikipedia “REST”. Retrieved from https://en.wikipedia.org/wiki/REST.
25. Open AI “Pricing”. Retrieved from https://openai.com/pricing.