A SUBSYSTEM OF THE INTEGRATION OF THE EDUCATIONAL PORTAL WITH THE LLM-BASE EDUCATIONAL CONTENT GENERATION SYSTEM

Підсистема інтеграції освітнього порталу з системою генерації освітнього контенту на базі LLM

by Oleksandr Ivchenko

Presented in Partial Fulfillment of the Requirements for the Degree

Master of Software Engineering

American University Kyiv

2024

APPROVED BY:

Sergiy Tytenko, Ph.D. Associate Professor

Abstract

LLM models have created a new era of learning and teaching methodologies. This article examines the profound impact of LLM models on the educational environment. The study explores their integration in various educational settings and highlights their universal application and transformative potential.

The essence of the analysis lies in the adaptability of OpenAI and ChatGPT in various educational and training scenarios. These technologies are tools and partners in learning, offering personalized, interactive, and engaging experiences. This allows the creation of educational materials and facilitates the teacher's work by providing instant feedback and support.

An essential aspect of this study is the availability and quality of education. LLM models have the potential to bridge gaps in education, provide an alternative perspective, generate high-quality resources, and provide exclusive education for people with different learning needs.

The practical implementation of comparing ChatGPT capabilities is illustrated using the example of generated content with an existing verified source. These examples demonstrate how educators integrate AI's power to improve curriculum development.

The study highlights the potential of combining technology and human experience to create an enriched learning ecosystem. The future of learning, shaped by artificial intelligence technologies, promises an improved educational experience and a more equitable and accessible world of knowledge and opportunity.

Contents

Chapter 1

1.1 Introduction

The modern world is rapidly evolving, and the educational landscape is transforming [1]. Modern technology, such as the ChatGPT language model, is said to have the power to change the educational system [2] completely. Traditional teaching methods are being challenged, and the need for innovative approaches is more apparent than ever. This statement is still relevant because the traditional learning environment of physical classrooms has remained unchanged [3].

Integrating artificial intelligence (AI) technologies becomes increasingly central in the quest for educational enhancement. It highlights the urgent need for innovative, tech-driven approaches in education, exploring the integration and application of artificial intelligence (AI) technologies in learning. This study begins with a deep dive into the evolving educational technology landscape, where traditional teaching methods are giving way to modern learning paradigms.

This exploration's core is a comprehensive examination of how OpenAI and ChatGPT can be integrated and adopted within educational contexts. The article delves into the current use cases of these technologies, showcasing their potential to enhance the quality and accessibility of education. This involves critically analyzing how AI technologies like ChatGPT can be harnessed to create more engaging, personalized, and compelling learning experiences.

Furthermore, the article presents real-world case studies that illustrate the practical application of ChatGPT in various educational settings. These case studies provide tangible evidence of the transformative impact of AI in education, offering insights into how educators and institutions are leveraging these technologies to reshape teaching methodologies and curriculum designs. These examples give us a clearer understanding of the potential challenges and benefits of integrating AI into the educational landscape.

This article aims to provide a thorough and nuanced understanding of the transformative potential of OpenAI and ChatGPT in education. Its goal is to show ways to responsibly and effectively integrate AI into educational systems to meet the needs of diverse learners. Doing so contributes to the broader discussion on the future of education, highlighting the critical role of technology in shaping a more adaptive and forward-thinking education.

1.2 OpenAI and ChatGPT in Education

The education landscape has significantly transformed in the 21st century, driven by swift technological advancements. The conventional teaching methodologies, typically characterized by uniform approaches, are progressively replaced by learning experiences more tailored and adaptive to individual needs. It has generated much buzz and dire predictions regarding student assessment in higher education and various other issues [4]. This transformation is primarily driven by integrating artificial intelligence (AI) and large language models, such as OpenAI's ChatGPT.

Traditional teaching often characterized a one-size-fits-all approach, where students received uniform lessons and assessments. In contrast, modern learning leverages the power of AI to provide personalized, dynamic, and interactive learning experiences. It adapts to each student's needs and preferences, making education more effective and engaging. This can empower learners to take charge of their learning and growth and develop the skills necessary for success as self-directed learners [5].

In the dynamic landscape of education, OpenAI and ChatGPT have emerged as transformative forces, influenced current practices and future possibilities and made significant contributions. Recent use cases encompass applications such as intelligent tutoring systems, automated assessment tools, instant feedback mechanisms, and real-world examples of how these technologies are deployed in classrooms and educational institutions.

The potential impact of OpenAI and ChatGPT in education is vast. These technologies promise to revolutionize traditional teaching methods, offering the potential for personalized, data-driven, and engaging learning experiences. We discuss the transformative possibilities of AI in education, examining how it can improve student performance, accessibility, and overall quality. [6] Moreover, pupils can receive automated performance feedback through ChatGPT [7]. For instance, a ChatGPT-based chatbot can provide students immediate feedback as they work through difficulties.

1.3 Integration LLM Content for Learning Enhancements

ChatGPT can be used to create intelligent tutoring programs that give students individualized learning experiences. An Intelligent Tutoring System (ITS) can track a student's development and modify the curriculum to suit their needs [8].

Case studies show how ChatGPT can track student progress, modify the curriculum to suit individual needs, and offer immediate assistance. Through practical implementations, it is shown how AI can enhance student performance and motivation. Research has revealed that although AI technology can partially supplant human labor, improving work and learning efficiency, it cannot fully replace human creative thinking skills [9]. Moreover, AI technology finds it challenging to replace complex human decision-making.

Work [10] explores the potential of ChatGPT as an educational tool, specifically in communication, business writing, and composition courses. The research uncovers opportunities and challenges associated with ChatGPT for both students and instructors. Findings reveal that ChatGPT offers students an integrated platform to seek answers to theoretical and application-based questions while allowing instructors to integrate technology into classrooms and facilitate discussions and evaluations of generated responses. However, it highlights challenges, notably the risk of students using ChatGPT unethically, potentially leading to diminished learning outcomes.

Additionally, the study suggests that while ChatGPT has the potential to replace search engines by providing quick and reliable information, it necessitates vigilant monitoring to prevent overreliance, urging instructors to adapt their assessments to ensure meaningful learning experiences. The study underscores the need for further research to comprehensively understand ChatGPT's implications for students and instructors.

The system for generating tailored reading exercises for middle school English learners in China demonstrates superior quality compared to human-written materials through integrations of extensive automated and manual evaluations. This system is designed to support teachers and students by utilizing Large Language Models (LLMs) to create reading comprehension exercises. The high quality of these exercises was validated by experienced English teachers, suggesting real-world potential. However, challenges persist in generating multiple-choice questions, particularly in crafting effective distractors, despite the system's use of ChatGPT [11].

This article [12] examines the impact of OpenAI Codex and similar tools on introductory programming education, particularly their use in generating programming exercises and code explanations. The study utilizes OpenAI Codex to create these resources, finding that a significant portion of the automatically generated content is novel and sensible. The study demonstrates how these tools can be influenced to align with programming concepts and contextual themes by supplying specific keywords and integration based on existing materials. The research emphasizes the value of large language models for instructors but highlights the need for quality oversight before delivering content to students. It also discusses the potential for future research to enhance the educational experience. Furthermore, the study explores how Codex can aid instructors in creating exercises and generating code explanations, with most exercises being sensible and novel. However, some adjustments may be required before integrating them into courses, and instructors can address these issues. The study also acknowledges that code explanations mainly cover the code but may contain minor inaccuracies that instructors or teaching assistants can quickly correct.

1.4 Balancing AI Utility with Student Privacy

The study by Lund and Wang [13] outlined the “advantages of ChatGPT, including bettering search and discovery, reference, and information services; cataloging and metadata generation; and content creation, as well as the ethical issues that need to be considered, like privacy and bias.”

As AI technologies gather and process student data, protecting privacy and adherence to data ethics become paramount when integrating learning portals. The ethical implications and regulations governing the use of student data, including GDPR and COPPA. GDPR in Europe and the Children's Online Privacy Protection Act (COPPA) in the United States require organizations to protect the personal data of individuals [14]. Additionally, the question is the complexities of maintaining individual privacy while extracting valuable insights through AI. By balancing data utilization and ethical responsibility, AI contributes positively to education.

AI-powered learning systems can have biases in the data they are trained on, affecting the quality of education—the investigated strategies to identify and improve quality, data preprocessing, and ongoing monitoring. AI systems reinforce and promote inclusivity, transparency, and fairness in education.

1.5 Future Prospects of LLM models for learners

The future of education based on integrating MLL models into learning portals holds immense potential for lifelong learners. AI technologies, including ChatGPT, are poised to offer continuous, personalized learning experiences. Lifelong learners can access knowledge tailored to their evolving interests and needs. We delve into microlearning, where bite-sized educational content is readily available based on the integration of LLM models using existing materials of learning portals, allowing individuals to upskill, reskill, and pursue their passions throughout their lives. [15] the authors assert that AI and ChatGPT integration promise to improve student's learning outcomes, enhance educational resource distribution, and strengthen educational quality oversight. As computers can rapidly perform tasks like high-quality writing, technical responses, and programming, the increasing potency of AI is driving fundamental transformations in educational objectives to align with societal needs.

As AI integrates deeper into education, the relationship between humans and AI takes center stage. The concept of a collaborative frontier, where human educators work hand in hand with AI to provide a richer learning experience. However, this partnership presents challenges, including maintaining ethical AI use and ensuring that humans remain in the educational process.

This article [16] introduces a high-level solution to integrate ChatGPT into assessment generation. It also discusses related work, including concept maps for evaluating understanding and decomposing educational content into smaller, manageable units, providing a more organized and coherent approach to assessment generation. This granular approach is crucial for LLMs to work effectively in this context.

1.6 AI and LLM in Digital Learning: Tailored Educational Content Generation

Integrating educational platforms with large language models (LLMs) for content creation has become increasingly important in the evolving digital learning environment. This initiative is designed to utilize the capabilities of cutting-edge artificial intelligence to produce and distribute educational content that is both scalable and customized to meet various learning requirements. The essence of this approach is using AI-driven systems, like ChatGPT, to generate educational materials that align with the curricular and pedagogical norms of established educational frameworks.

The main goal of this project is a thorough assessment of the quality and pertinence of the content created by LLM-based systems. Through a comparison with traditional educational materials, the project aims to determine the efficacy of AI-produced content and achieve learning goals. This evaluation is concerned not only with the factual accuracy of the content but also with its capacity to engage students and aid the learning process across different levels.

Technology is essential in the modern era of digital learning in shaping the educational experience. Conventional teaching and learning methods are being enhanced and, in some cases, replaced by technological innovations that bring about more interactive, captivating, and customized learning journeys. The adoption of LLM-based content generation within educational platforms symbolizes this transformation. It signifies a progressive approach to education, where AI's potential is tapped to supplement and enrich both teaching and learning activities.

This project investigates how technological advancements, especially in AI and machine learning, can revolutionize how educational content is formulated and disseminated. It explores AI's potential in making education more approachable, efficient, and tailored to the diverse requirements of students. Automating the content generation and customization to fit specific educational outcomes promises a more adaptable and enjoyable learning experience.

Furthermore, this project seeks to establish technology as a vital educational partner. It envisions enabling educators to concentrate on more crucial teaching aspects, like nurturing critical thinking and creativity, while AI manages the routine aspects of content generation. Thus, technology is perceived not merely as a tool for enhancing educational practices but as a catalyst for redefining traditional educational models, leading the way toward a future where learning is more inclusive, stimulating, and effective.

1.7 Advancing Education through LLM Integration: Balancing Personalization with Content Accuracy and Consistency

Today's learners come from diverse backgrounds and possess varied learning styles and paces. This diversity demands a shift towards personalized and adaptive educational content, which can

cater to individual learning preferences and needs. Personalized learning focuses on tailoring educational experiences to suit individual strengths and weaknesses, while adaptive learning dynamically adjusts content based on a student's real-time performance and progress.

This shift towards personalized and adaptive content reflects a broader transition from teacher-centric to learner-centric models in education. Such models prioritize the learner's engagement and interaction with content, recognizing that effective learning depends on how relevant and resonant the material is for the individual student.

Incorporating large language models (LLMs) such as ChatGPT into educational portals marks a significant shift in the educational landscape. These AI-powered systems leverage their advanced language processing capabilities to produce content that is not only rich and varied but also customized to align with specific learning goals and preferences. Through the analysis of extensive educational data, LLMs can discern learning trends and adaptively generate content that caters to the unique needs of each learner.

Such integration enables a more interactive and responsive educational experience. For example, students facing challenges with specific topics can receive customized assistance and resources from the LLM system. At the same time, more proficient learners can be provided with content based on existing materials from learning portals that stretch their abilities and keep them engaged.

Despite the promise of LLM-based systems, they are not without challenges, particularly concerning the accuracy and educational relevance of generated content. AI systems, for all their sophistication, can occasionally produce content that may be inaccurate, outdated, or misaligned with specific curricular requirements. In education, where accuracy is paramount, such discrepancies can harm learning.

Addressing these challenges requires a robust system of expert oversight and user feedback. Educators and subject matter experts are crucial in periodically reviewing AI-generated content to ensure its factual correctness and curriculum alignment. Additionally, feedback from both educators and learners is invaluable in assessing the effectiveness and appropriateness of the content, guiding further refinement of the AI system's outputs.

Implementing a structured system for ongoing review and feedback is essential. Such a system enhances the AI's output quality and ensures the content remains relevant, accurate, and effective for the target learners. By tackling these challenges, integrating LLM-based systems within educational portals can significantly transform and enrich the educational experience in our digital age.

Conclusion

The exploration of the intersection of artificial intelligence and education, especially with a focus on the transformative potential of OpenAI and ChatGPT, began with a deep dive into the evolving edtech landscape, where modern learning paradigms are reshaping traditional teaching methods,

keeping in mind the challenges faced by the education sector and the promising opportunities presented artificial intelligence technologies.

1. The analysis identified current use cases and the potential impact of artificial intelligence technologies on improving the quality and accessibility of education.

2. Case studies and practical implementations demonstrated how ChatGPT is actively shaping the educational landscape, demonstrating the tangible benefits of AI. The potential of artificial intelligence will accompany students throughout their lives.

3. Education demonstrates the incredible potential of technology to improve learning, provide personalized support, and make knowledge more accessible.

4. Revolutionary advances in personalized and adaptable learning experiences incorporate cutting-edge Large Language Models (LLMs) such as ChatGPT. These AI-driven systems can tailor educational content to individual needs, offering a more engaging and hands-on learning experience. However, realizing this potential requires careful attention to the accuracy and educational relevance of AI-generated content. Collaboration between educators and subject matter experts in analyzing and refining AI results, coupled with constructive feedback from users, is essential to overcome problems of consistency and relevance. As we overcome these obstacles, the consolidation of LLMs in education promises to redefine the learning environment to be more personalized, flexible, and responsive to the diverse needs of today's students. This paradigm shift improves the educational experience and more effectively prepares students for the challenges of an ever-changing world.

5. The future of education is promising for empowering people to realize their potential. But is technology that good? For this purpose, a tool was created to answer this question by comparing information from a verified source and the context generated by ChatGPT.

Chapter 2

2.1 System Architecture and Design of Integrated Learning System

The Integrated Learning System (ILS) is an integrated platform designed to facilitate educational content creation, management, and distribution. The system interfaces with the LLM-based educational content generation system to supplement traditional educational materials with dynamically created content. The Integrated Learning System is envisioned as a robust and scalable system, leveraging cutting-edge AI technology to enhance educational content. Its focus on flexibility, quality control, and ease of use aims to provide an enriching experience for administrators and learners.

The Integrated Learning System user interface is a web gateway designed to give administrators complete control over creating and managing educational content. It is designed to provide a simple and intuitive experience, ensuring administrative tasks such as course creation, content editing and regeneration, and quality control of generated material are simple and efficient. The user interface, characterized by responsiveness and clarity, allows administrators to quickly navigate various functions, from downloading content to viewing and organizing course materials. This interface makes it easy for users to access courses, creating an engaging and interactive learning environment.

The heart of the eLearning portal is its robust backend, based on Python and the Django framework. This backend handles the portal's core business logic, including data processing, user authentication, and maintaining the overall application workflow. The Django platform provides efficient data retrieval and content creation processes, ensuring a smooth and reliable system operation. In addition, the backend handles API interactions, which are critical for integrating external resources and services into the portal and enhancing its capabilities.

The portal uses SQLite database, a lightweight yet powerful storage solution for data management. This database efficiently stores important information such as course metadata, user profiles, and educational content. The choice of SQLite reflects the system's need for a database that is not only reliable and fast but also easy to maintain and manage. This database structure ensures that data retrieval and storage processes are optimized for performance, contributing to the eLearning portal's overall responsiveness.

The most important feature of an e-learning portal is its level of integration, which establishes connections with external content-creation services and educational resources. This level includes interfaces to the LLM-based educational content generation system and the semantic portal, allowing the portal to access various dynamic educational content created by artificial intelligence and structured academic resources. Integration with the LLM-based educational content generation system allows the portal to expand course offerings with content managed by artificial intelligence, which makes the learning process more diverse and adaptive. Meanwhile, connecting to the

Semantic Portal allows you to include various educational materials in the portal database, enriching the course content.

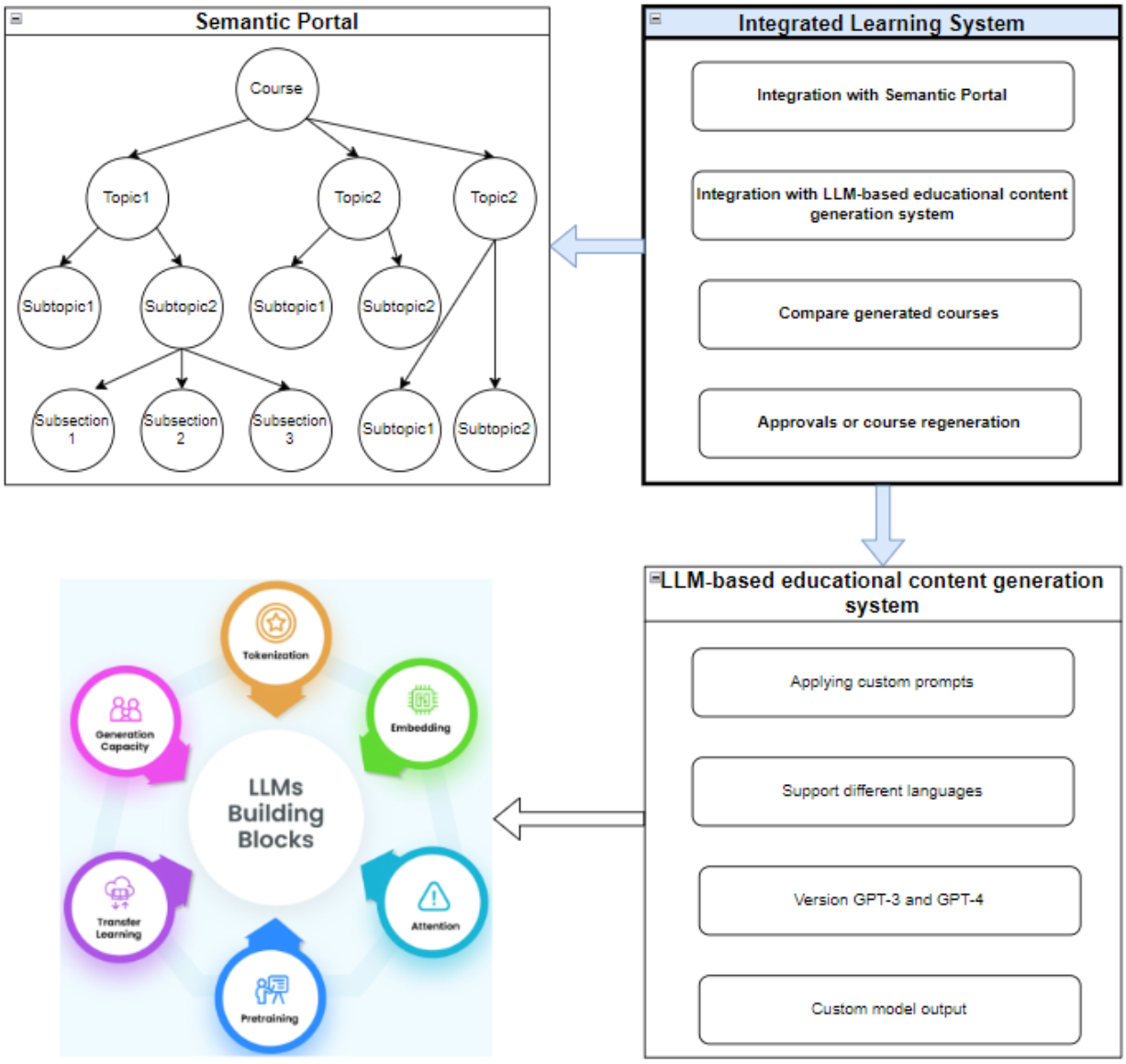


Figure 2.1 System Architecture Overview of Integrated Learning System

Functional Capabilities:

Name	Description
Content Scraping and Storage	Automated scraping of educational content from predefined web sources. Persistent storage of scraped data in a structured format within the SQLite database.
Course Creation and Management	Tools for administrators to select and assemble course structures from available content.

	Content generation capabilities using AI models from the LLM-based educational content generation system can create basic to advanced course materials.
Quality Assurance	Mechanisms for administrators to review, approve, or reject AI-generated content. Features for comparing traditional and AI-generated content to ensure quality and relevance. Interface for end-users to access and compare course materials.

Table 2.1 Functional Capabilities

Technical Specifications:

Name	Description
Frontend	Responsive design is compatible with various devices and screen sizes.
Backend	Language: Python 3.x. Framework: Django 3.x leverages the Django REST framework for API construction.
Database	SQLite for development and testing, with the option to scale to PostgreSQL for production. ORM: Django's built-in ORM for database interactions.
External API Integration	RESTful API endpoints for interacting with the Semantic Portal for content scraping. RESTful API endpoints for interacting with the LLM-based educational content generation system to create course materials with personalized content.
Security	Regular security audits and access control to certain functionality only for the administrator.
Deployment	Support deployment on cloud platforms such as AWS, GCP, or Azure.
Documentation and Support	System documentation for admin.

Table 2.2 Technical Specifications

The Integrated Learning System (ILS) embodies modern educational technologies, skillfully combining the simplicity and efficiency of the user interface with reliability. ILS is based on the Python and Django framework. The user interface is designed to provide easy navigation and content management, making administrative tasks more manageable.

At the core of the user interface is a powerful backend that efficiently handles data, user authentication, and API interactions. Using Django not only ensures the smooth operation of the system but also ensures smooth integration with external resources, expanding the capabilities of the portal. The SQLite server database is a strategic choice that reflects the system's need for a

reliable, easily managed storage solution. This database reliably supports the portal's core functions, from storing course metadata and user profiles to managing educational content.

The most distinctive feature of ILS - the level of integration - facilitates communication with external content generation services, such as the LLM-based educational content generation system, and educational resources, such as the Semantic Portal. This integration is critical to diversifying educational content and making learning more adaptable and inclusive. Technical capabilities include content collection, course creation, and quality assurance mechanisms. These features ensure that the content is high quality but also relevant. These features ensure that the content is high quality, yet relevant, delivering a comprehensive and dynamic educational experience backed by a strong technical foundation and user-centric design. The ILS highlights the critical role of technology in shaping the future of learning.

2.2 Subsystem Design for LLM-Based Content Generation in Educational Portals

The application's architecture is designed for scalable and easy navigation through its content. The backend services, written in Python using the Django framework, are robust and reliable, handling essential operations like business logic execution, data retrieval, content generation, and API interactions. The system's integration layer connects to external services to enrich the educational content. Design supporting expansion and integration with various content generation services, including LLM-based educational content generation system like the Svitoch System and educational resources like the Semantic Portal.

Each template within the template’s directory serves a specialized function, ensuring users have a seamless and interactive experience.

With its meticulously organized directory structure, the Integrated Learning System is a comprehensive platform for checking generated content quality and delivering programming courses. The project is structured to facilitate the creation, distribution, and management of educational content through a website.

Migrations: Database schema changes are tracked here, allowing for robust data model evolution over time.

db.sqlite3: This file is the SQLite database where all data related to the courses, user profiles, and system operations are stored, providing a lightweight and easy-to-manage persistence solution.

manage.py: A command-line utility that lets you interact with this Django project in various ways. The tool triggers administrative tasks such as migrations, server startup, and other operational needs.

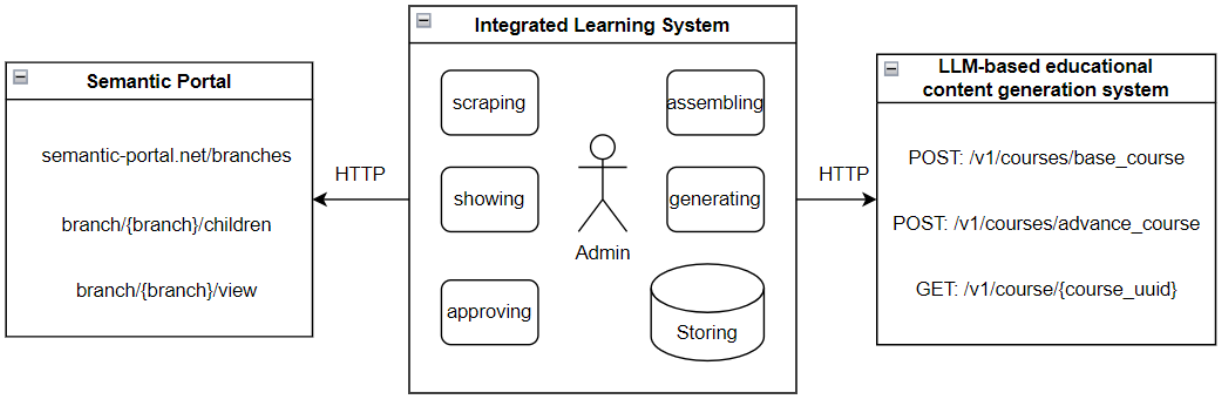


Figure 2.2 Functional diagram

The navigational guide for HTTP requests serves the urls.py file, matching URLs to their corresponding view functions. This setup forms the backbone of user interaction within the application, ensuring that each URL directs the user to the correct page and function. Each URL pattern facilitates easy navigation and logical flow within the eLearning portal, simplifying how administrators and users interact with the system. The organization of these URL mappings underscores the system's commitment to providing a user-friendly experience.

A collection of view functions that define the business logic for various operations users can perform on the platform serves the views.py file. Each function is tied to a specific URL pattern, responding to user requests and rendering the appropriate HTML templates with context data. Each view function interacts with the Django ORM to retrieve, process, and save data, demonstrating the system's reliance on a robust backend framework to handle complex educational content management tasks. The seamless interaction between the frontend templates and backend logic encapsulates the dynamic nature of modern web applications, where server-side processing results in real-time user interface updates.

The utility module that encapsulates various functions essential for the system's operation is utils.py file. These functions are designed to interact with external resources, handle data processing, and support the application's backend services. This utility module is a critical part of the Integrated Learning System backend, responsible for interfacing with external educational content sources and organizing that content into a format suitable for the application's needs. It showcases using Python's robust libraries for web scraping, HTTP requests, and JSON manipulation to maintain an up-to-date and rich educational content repository. The module includes error handling to catch exceptions from API requests or database operations and printing error messages to the console for debugging purposes.The **models.py** file within the Django framework defines the data structure for the eLearning portal. The ORM (Object-Relational Mapping) layer facilitates interaction with the database using Python classes. In essence, models.py stores and organizes the data that powers the portal's educational experiences.

The form structures used within the eLearning portal defines the forms.py file in the Django application. These forms serve as the interface for collecting user data, whether for logging in, selecting course topics, or writing new course content. The forms are thoughtfully integrated with the backend services, reflecting a seamless user experience and robust data handling, providing structured and styled inputs for data that the system needs to process various requests and operations. By utilizing Django's form ensures data validation and efficient handling of user inputs, which are crucial for maintaining the integrity and functionality of the system. The forms are rendered in templates and processed in view functions.

The Django template system, which includes a collection of custom template tags and filters designed to assist with data retrieval and presentation logic within the templates of the eLearning portal is part of the branch_tags.py file. The module exemplifies how Django's template language can be extended with custom functionality to suit the specific needs of an application, which requires dynamic data retrieval and presentation based on user interactions and system state.

These template tags and filters are custom utilities that allow the templates to manipulate and display data dynamically. They abstract complex queries and data manipulation away from the templates, simplifying their code and enhancing readability and maintainability. Using these tags and filters, template developers can perform common operations on data objects without writing verbose Python code within the templates. This separation of concerns makes the templates cleaner and focuses on presentation logic, while complex data handling is managed within the branch_tags.py module.

The provided HTML template files, collectively, form the user interface of the eLearning portal, each serving specific roles to present data and interact with the user within a Django web application. These templates deliver a comprehensive user experience, from content selection and course creation to authentication and data presentation. The eLearning portal presents a well-structured and interactive platform by leveraging Django's templating system. The foundational layout of the portal, base.html, sets the basic structure shared across all templates. It includes Bootstrap for responsive design, ensuring the portal is accessible on various devices, a consistent navigation bar that adapts based on user authentication status, and directing to different application parts. A {% block title %} for dynamic page titles, and a {% block content %} placeholder, where specific page content will be injected by other templates extending this base.

The described application utilizes Django, a Python-based framework, in various aspects of its architecture, demonstrating Django's suitability and effectiveness. The framework's robustness is evident in handling backend services, including business logic, data operations, and API interactions. Django's organized directory structure supports content quality checks and course delivery efficiently. Its ORM layer, form handling, and template system enable seamless data management and user interface dynamics. The application leverages Django's features for scalable, interactive eLearning experiences, highlighting Django's versatility and strength in managing

complex web applications with diverse functionalities. Django is excellent for this capstone project.

2.3 User and Admin capabilities

The system caters to two primary user roles: the regular user (learner) and the admin (educator or course creator).

User: Users have access to a range of courses generated from various sources:

- Courses directly taken from the semantic portal in whole or in part.
- Courses are completely created by the administrator on a custom topic.
- Basic courses generated by ChatGPT, designed for foundational understanding.
- Advanced courses generated by ChatGPT offer an in-depth exploration of topics.

Admin: Creation of Courses: Admins can create courses using tree primary sources and opportunity to approve or regenerate course

1. The semantic portal's public API provides a repository of educational topics.

2. The Svitoch project API facilitates the generation of basic and advanced courses using ChatGPT.

3. Creation of an individual course. Administrators can set topics to generate courses using a list of required topics.

- Course Approval: Admins can approve generated courses, making them available to users.
- Course Regeneration: Admins can regenerate courses based on feedback or updated information to ensure content relevance and accuracy.

The Use Cases diagram illustrates the roles and capabilities assigned to each actor within the system, clearly representing the system's functionality from the perspectives of end-user engagement and administrative control.

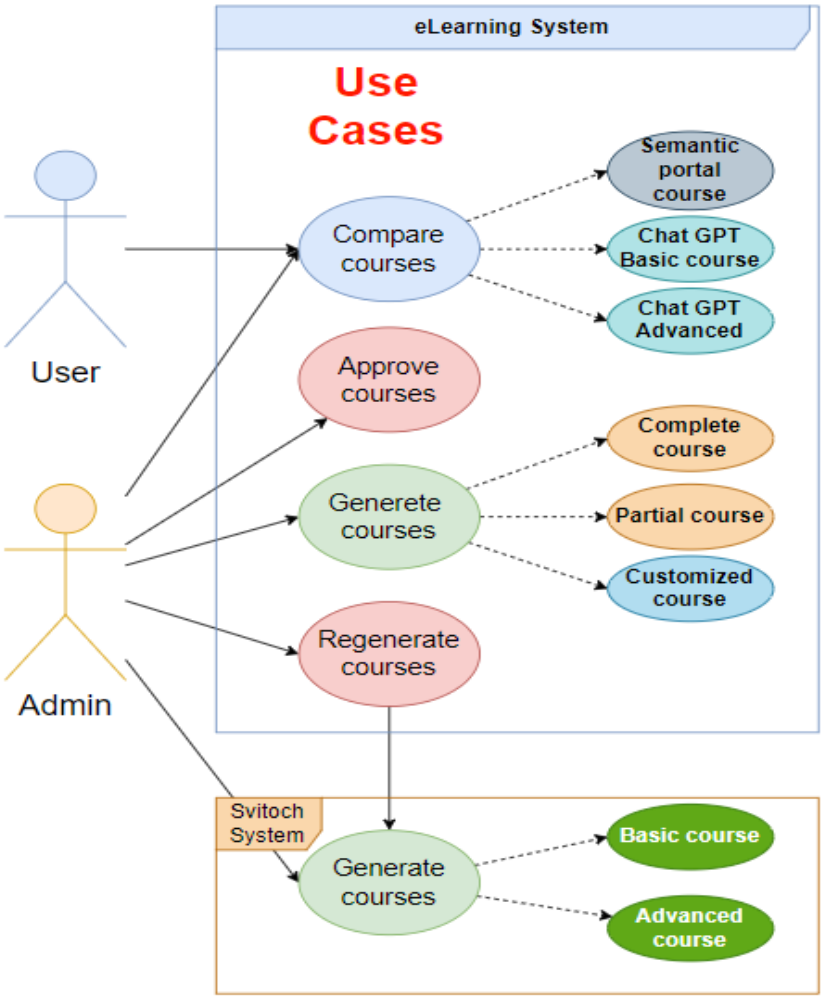


Figure 2.3 Use cases diagram

User Interactions:

The User, typically a learner or a course participant, can access a streamlined set of functionalities. The primary interaction for the User is the ability to "Compare courses," which involves juxtaposing the course content sourced from the Semantic portal with the content generated by ChatGPT, both in its basic and advanced forms. This comparison is crucial in helping the User to discern the quality and depth of the content, providing a means to evaluate the learning material's effectiveness.

Admin Capabilities:

The admin, who manages the course content and oversees its quality, has a more extensive set of interactions with the system:

Approve Courses: The admin can approve courses, an essential step that marks the content as verified and suitable for the User's consumption. Approval is contingent upon the admin’s assessment of the content's quality and relevance.

Generate Courses: The admin can initiate the generation of new course materials. This process taps into two sources: the Semantic portal for existing course structures and the Svitoch system for creating course content through ChatGPT. This dual-source generation allows for the production of complete, partial, or customized courses, catering to the diverse needs of the Users.

Regenerate Courses: In cases where course content does not meet the required standards or needs to be updated, the admin can trigger a regeneration process. This process revisits the course generation mechanism, applying modifications or updates as necessary to enhance the course quality.

LLM-based educational content generation system Integration:

At the core of the admin’s interactions is the Svitoch system, which is responsible for "Generate courses". This function is the backbone of the system's content creation capabilities, employing ChatGPT to produce two types of courses:

Basic Course: Generated with foundational content in mind, suitable for users seeking to grasp the fundamental concepts of a subject.

Advanced Course: Crafted to provide an in-depth exploration of topics for users who require more comprehensive knowledge.

The User's ability to compare courses is directly influenced by the admin’s decisions to approve, generate, or regenerate courses, ensuring that the User always has access to high-quality and relevant content. The admin’s role is pivotal as the gatekeeper for content quality and course availability. Through the Admin's interactions with the Svitoch system, the Integrated Learning System ensures that the content is generated and aligned with educational objectives.

The generation of courses through the Svitoch system showcases the system's reliance on sophisticated AI to meet the demands of modern educational content delivery. It emphasizes the system's commitment to quality and adaptability in educational content delivery.

2.4 Integration with the Semantic Portal

A semantic portal is a web resource that uses semantic technologies to organize and present information with a rich description of different programming languages. These technologies may include knowledge graphs, ontologies, and natural language processing to provide more intelligent and context-sensitive search and navigation capabilities.

This resource integrates easily using the public Semantic Portal API to leverage a variety of programming language resources. HTTP requests are used to retrieve information about various

programming languages. The semantic portal has extensive and carefully organized content covering a wide range of programming languages. The resulting data includes best practices, coding examples and advanced concepts and covers a wide range of content, making it a one-stop solution providing accurate and up-to-date content that is always in sync with current industry standards and practices.

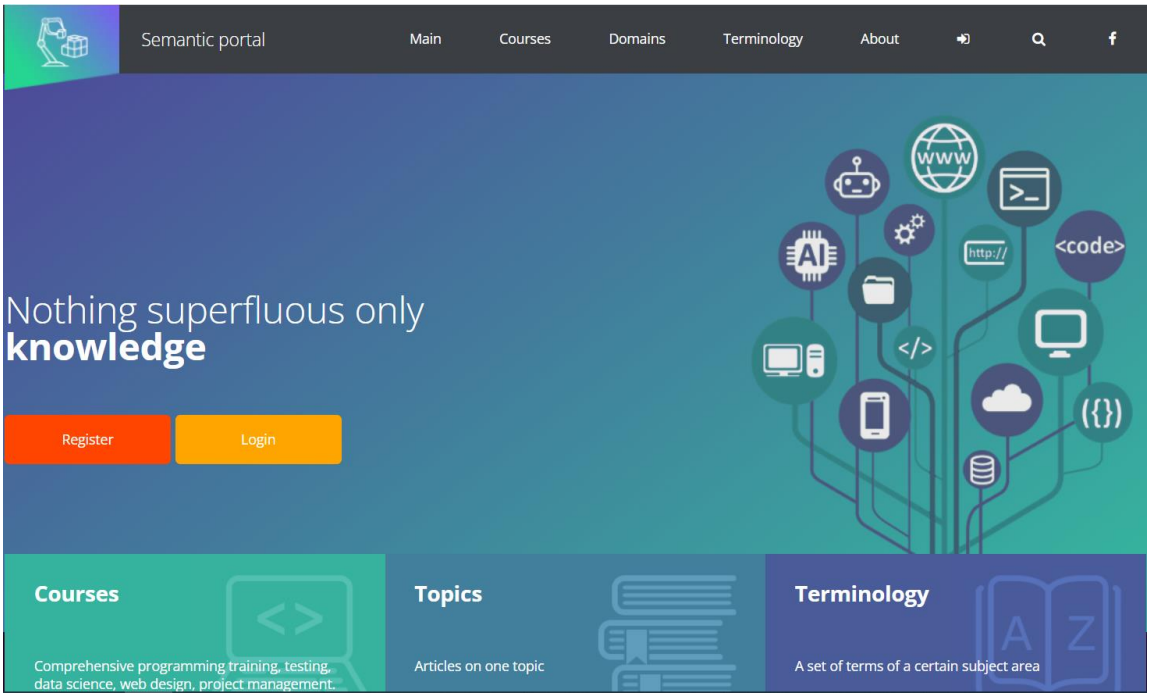


Figure 2.4 Link to resource http://semantic-portal.net/

Through integration with Semantic Portal, this project gains access to high-quality programming education. As the Semantic Portal updates its content, the project immediately reflects these changes, ensuring students have access to the latest information and trends in programming languages.

Integration of the project with the public API of the semantic portal allows obtaining course data as part of a multi-step process.

Step 1. Collection of course names

The system initially connects to the Semantic portal to fetch all available courses.

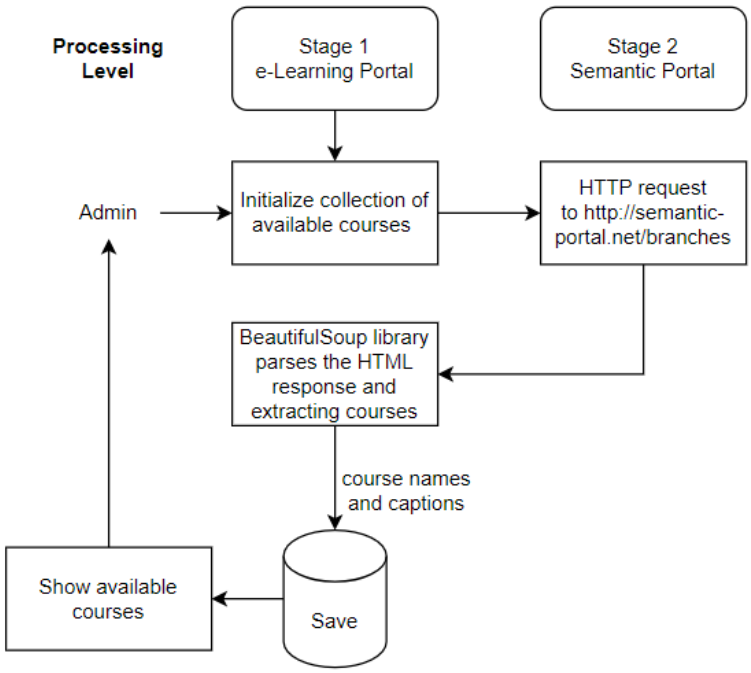


Figure 2.4.1 Process of collection of course names

Start point decorated with @login_required, ensuring that only authenticated users can invoke this function.

1. Requests the Semantic Portal to scrape data on available educational branches.

2. It retrieves course names and captions using the requests library and BeautifulSoup for parsing HTML.

```
1 usage   Alex Ivchenko
def get_data_from_server_and_save():
    url = 'http://semantic-portal.net/branches'
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58

    response = requests.get(url, headers=headers)
    global branch
    branch = ''

    if response.status_code == 200:
        soup = BeautifulSoup(response.content, features: 'html.parser')
        links = soup.find_all( name: 'a', class_='caption')
        for link in links:
            href = link.get('href')
            branch = href.split('/')[-1]
            caption = link.find('h3').text.strip()
            save_branchList(branch, caption)
    else:
        print(f"Failed to retrieve the page. Status code: {response.status_code}")
```

Figure 2.4.2 Fetch all available courses

3. Saving the data to the BranchList model in the database, creating a structured representation of the available courses

```
1 usage   Alex Ivchenko
@register.simple_tag()
def save_branchList(branchSlug, caption):
    if not BranchList.objects.filter(branch_name=branchSlug).exists():
        branch_instance = BranchList(branch_name=branchSlug, branch_caption=caption)
        branch_instance.save()
```

Figure 2.4.3 Save available courses

4. Providing a form-based interface populating a dropdown menu with available courses.

5. This endpoint manages the submission of user-selected a course. It's where the application processes, which parts of the curriculum will be developed or updated.

Step 2 Course Subpage Gathering

The system performs an API call upon course selection to gather all topics and subtopics related to the chosen course. Recursion navigates the course's structure, ensuring a comprehensive collection of all related educational materials.

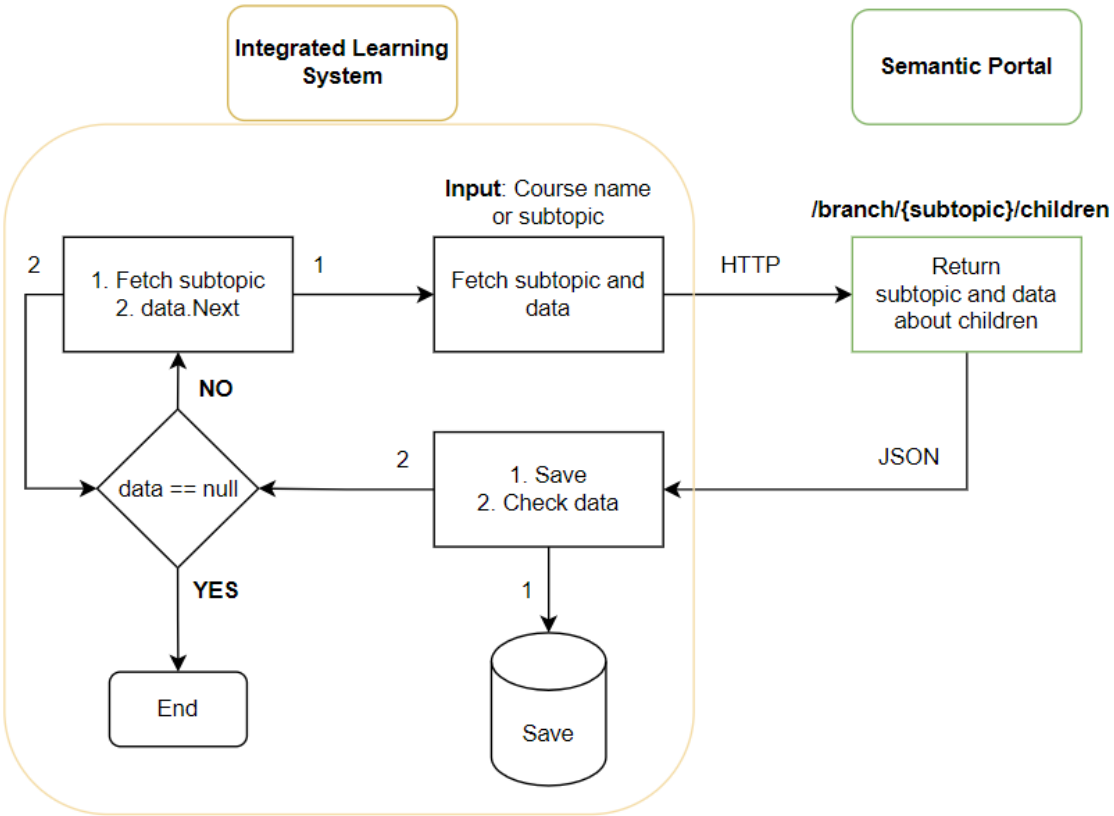


Figure 2.4.4 Process of gathering all topics

1. The administrator selects a course to obtain detailed data and subtopics for the selected course and clicks the submit button.

2. It checks whether the course data exists in the database; if not, then a POST request is sent to the semantic portal to obtain information.

3. The recursive function interacts with the Semantic Portal API to retrieve the children of a given branch and store the data in the database. This ensures that all subtopics and related content are captured and saved.

```
2 usages  Alex Ivchenko *
def fetch_children_recursive_and_save(branch_name):
    api_url = f'{semantic_portal}/branch/{branch_name}/children'
    headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    try:
        timeout_seconds = 10
        response = requests.get(api_url, headers=headers, timeout=timeout_seconds)
        response.raise_for_status()  # Raise an exception for HTTP errors
        data = response.json()
        global branch
        save_requested_branches(branch, branch_name, data)

        # Fetch children recursively
        for child in data:
            fetch_children_recursive_and_save(child['view'])
    except requests.RequestException as e:
        print(f"Error fetching data from API: {e}")
```

Figure 2.4.5 Fetch function for gather all topics

4.This data is stored in the BranchRequest model, which links to the BranchList and holds JSON fields for both data and view, storing structured information about the course content.

```
6 usages  Alex Ivchenko
class BranchRequest(models.Model):
    course = models.ForeignKey( to: 'BranchList', on_delete=models.SET_NULL, null=True)
    branch = models.CharField(max_length=255, unique=True)
    data = models.JSONField()
    view = models.JSONField(null=True)

    Alex Ivchenko
    def __str__(self):
        return self.branch
```

Figure 2.4.6 BranchRequest model

5. The results display confirms to the user what has been selected and saved and provides navigation options for next steps.

6. User interface for creating personalized course content. It extends the base template and uses JavaScript for select all and deselect functionality, improving the user experience. The data is displayed on a form that dynamically generates a ModelMultipleChoiceField to select multiple branches. The __init__ method configures the form's initialization by setting up a set of queries to retrieve a specific BranchList object by its name, and then retrieves any associated BranchRequest objects, which may contain additional data or queries associated with the branch. If the branch does not exist, it handles the exception gracefully and returns None. All actions are automated and based on user actions or choices made earlier in their session. The widget for the branch field is a CheckboxSelectMultiple, allowing users to select multiple branches at once, with additional styling applied using HTML attributes.

Step 3 Course Content Assembly

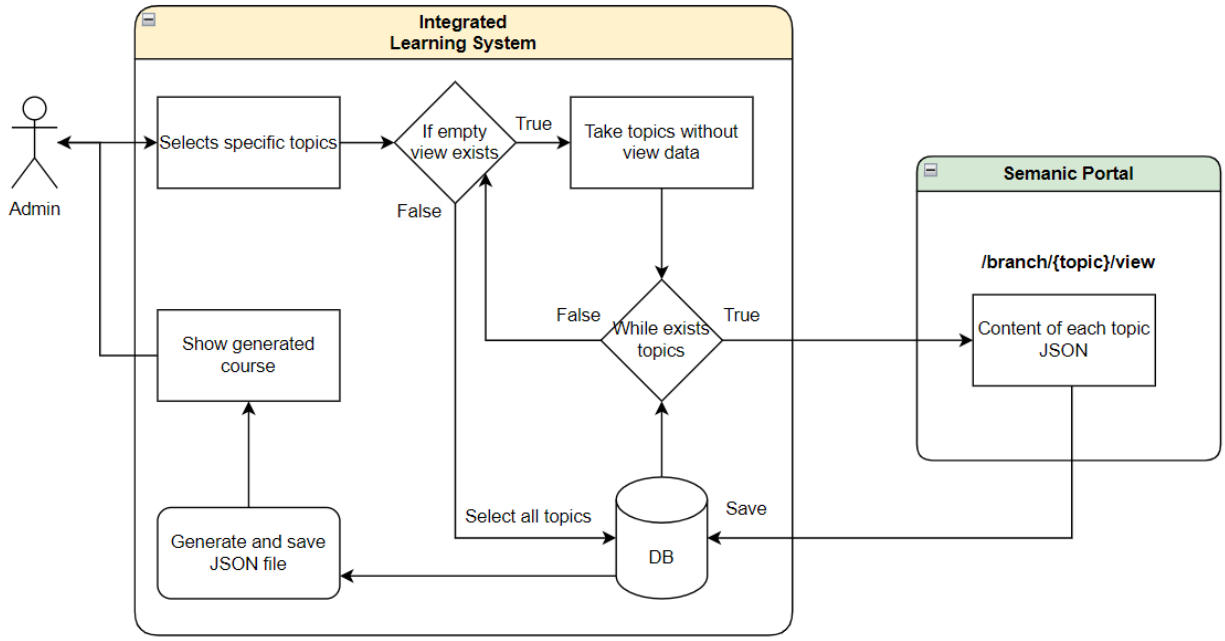


Figure 2.4.7 Process of collecting all content for course

1. The administrator selects specific branches on the semantic portal for inclusion in the course content and clicks submit.

2. The function processes the list of branches that do not have detailed views in the BranchRequest (view__isnull=True)

3. Using the loop to retrieve all view details for a specific branch from the Semantic Portal API and return the data.

```
1 usage   Alex Ivchenko
def fetch_empty_views_and_save(items):
    data_list = []
    for key, value in items.items():
        for branch_obj in value:
            data_list.append(branch_obj)

    branches_without_view = get_requested_branch_text(data_list, view=False)

    for item in branches_without_view:
        item.view = fetch_branch_view(item.branch)
        item.save()

    return generate_json_file(data_list)
```

Figure 2.4.8 Fetch function for saving views data

4. Saves data to view column in BranchRequest

5. Generating a JSON file compatible with the Svitoch project interface.

```
1 usage   Alex Ivchenko
def init_json():
    global branch
    initBranch = get_branch_text(branch)
    result_json = serialize( format: 'json', initBranch)
    result_data = json.loads(result_json)

    if result_data[0]['fields']['view'] is None:
        view = result_data[0]['fields']['data'][0]['view']
    else:
        view = result_data[0]['fields']['view']['caption']
    contex = result_data[0]['fields']['data'][0]['caption']

    resultJSON.append({
        "query": view,
        "llm_version": "gpt_4",
        "language": "English",
        "course_content": {
        "course_name": contex,
        "lessons": [],
        "content": []
        }
    })
```

Figure 2.4.9 Init JSON file for API

5. Display course content obtained from the semantic portal. And it has a structure: a sidebar with headings from the course content that acts as an interactive table of contents. The main content area displays course information with smooth scrolling for better navigation.

Figure 2.4.10 Generated course Example

2.5 Integration with the LLM-based educational content generation system

Selected topics are subjected to another layer of API interaction with the LLM-based educational content generation system through the Svitoch project, prompting ChatGPT to generate relevant content. The Svitoch project API is a bridge to ChatGPT that helps AI create basic and advanced versions of course content.

This project is based on the Python programming language as it extensively supports large language models and related technologies such as artificial intelligence, natural language processing, and machine learning. An essential feature of the project is direct integration with the OpenAI API, which serves as the basis for creating training materials and courses. The API allows the system to leverage the power of advanced natural language processing models developed by Open AI, offering resources for machine learning algorithms and language models. The backend part uses the LangChain Framework to optimize work with the Open AI API, the primary tool for generating materials. Additional options include choosing between GPT 3 or 4 versions. It is important to note that selecting these two options can significantly impact the quality and cost of the materials created. For example, GPT 4, although it provides more detailed and detailed answers, is more expensive.

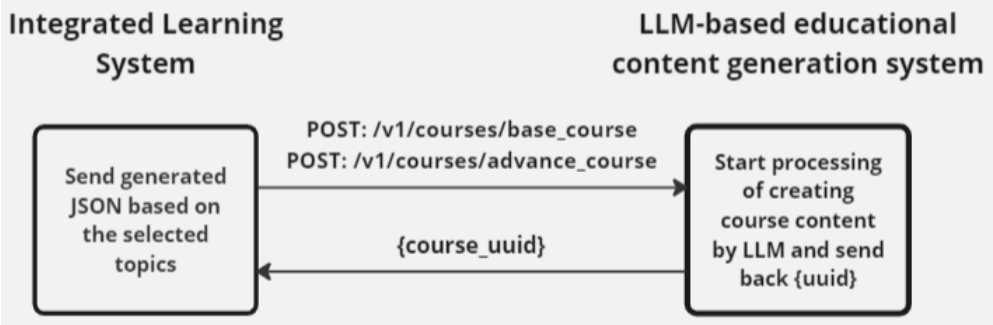


Figure 2.5 API calls to LLM-based educational content generation system

Once the administrator selects the desired topics, the eLearning portal generates the formatted structure and content of the selected topics into a JSON file. Available URLs for generating courses:

- POST: /v1/courses/base_course

- POST: /v1/courses/advance_course

Although the Svitoch project supports GPT-3 and GPT-4, it has been agreed that by default, GPT-3 is used to generate basic courses (/v1/courses/base_course), and GPT-4 is used to create advanced courses (/v1/courses/advance_course)

The structure of the JSON file must correspond to a strictly defined interface that supports the Svitoch project, and have the form.

```
{
  "query": "ASP.NET Core",
  "llm_version": "gpt_4",
  "language": "English",
  "course_content": {
    "course_name": "A tour of the ASP.NET Core language",
    "lessons": [
      {
        "title": "Introduction to ASP.NET Core and MVC Architecture",
        "topics": []
      },
      {
        "title": "Building Web Applications with Razor Pages",
        "topics": []
      },
      {
        "title": "Working with Data: Entity Framework Core",
        "topics": []
      },
      {
        "title": "API Development and RESTful Services",
        "topics": []
      },
      {
        "title": "Advanced Topics and Best Practices",
        "topics": []
      }
    ],
    "content": {}
  }
}
```

Figure 2.5.1 JSON structure that supported by LLM-based educational content generation system

The JSON is passed to ChatGPT via the Svitoch project, which returns a generated course based on the entered topics in JSON format. The generated data is obtained using the returned universally unique identifier (UUID) for each course. Using this identifier, the system can obtain information

about the generation status of a particular course, as well as receive an already generated course. Available URL GET:/v1/course/{course_uuid}.

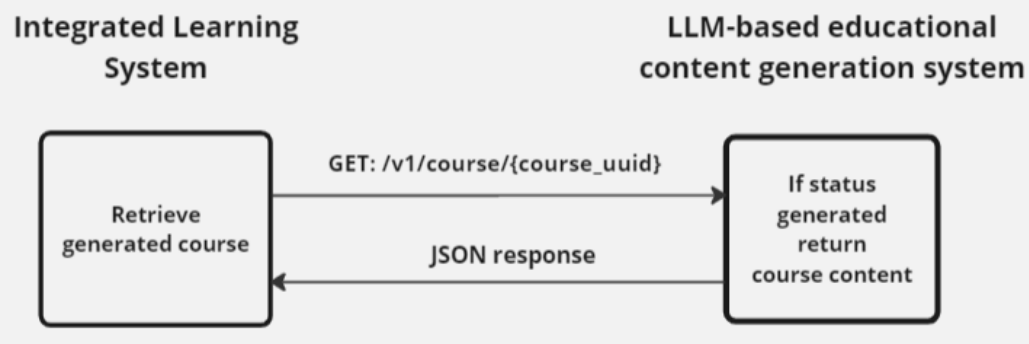


Figure 2.5.2 Process of receiving the generated course

There are three possible response statuses:

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

The returned JSON contains all information about the topics and generated content in Markdown format. Markdown format is widely supported, allowing users to utilize the materials without additional processing or formatting.

This newly created content is then stored in the system's database.

It should be highlighted that the quality of materials produced differs based on the specific GPT model employed in the generation process. Key variations are observed in the intricacies of course content and coding illustrations. GPT-4, for instance, provides more comprehensive explanations and samples than its predecessor, GPT-3. However, it's essential to recognize that the richness and accuracy of the generated content are enhanced by providing more elaborate descriptions of lessons and subjects. The level of detail also affects the generation speed. And it is important to note that generating materials can take up to 5 minutes. The cost of the request increases depending on the number of expected materials. The cost of creating Advanced courses using GPT-4 is quite expensive at the moment. Although the cost of GPT-3 is lower, the quality of the resulting materials is also lower, although GPT-4 also does not always do the job well.

Cost creating a 20+ page course:

Progress in C#: GPT 4 costs $4.14,
C# Basic: GPT 3 costs $0.27.

2.6 Course Creation Workflow Analysis

Course Creation Workflow displayed on a sequence diagram effectively communicates the interaction between the eLearning System, the Semantic Portal, and the Svitoch system. Each system plays a critical role in the course creation process:

- The Semantic Portal serves as the content source, providing the raw material for course topics and subtopics.
- The Svitoch system acts as the content generation engine, leveraging the power of GPT to create enriched and contextually relevant course materials.
- The eLearning System is the orchestrator, facilitating the course creation through its user interface and backend services.

The sequence diagram illustrates the comprehensive workflow that an Administrator follows to create and update course content within the eLearning System. This workflow is a critical aspect of the system's backend operations, ensuring that the course offerings are current, comprehensive, and tailored to the needs of learners.

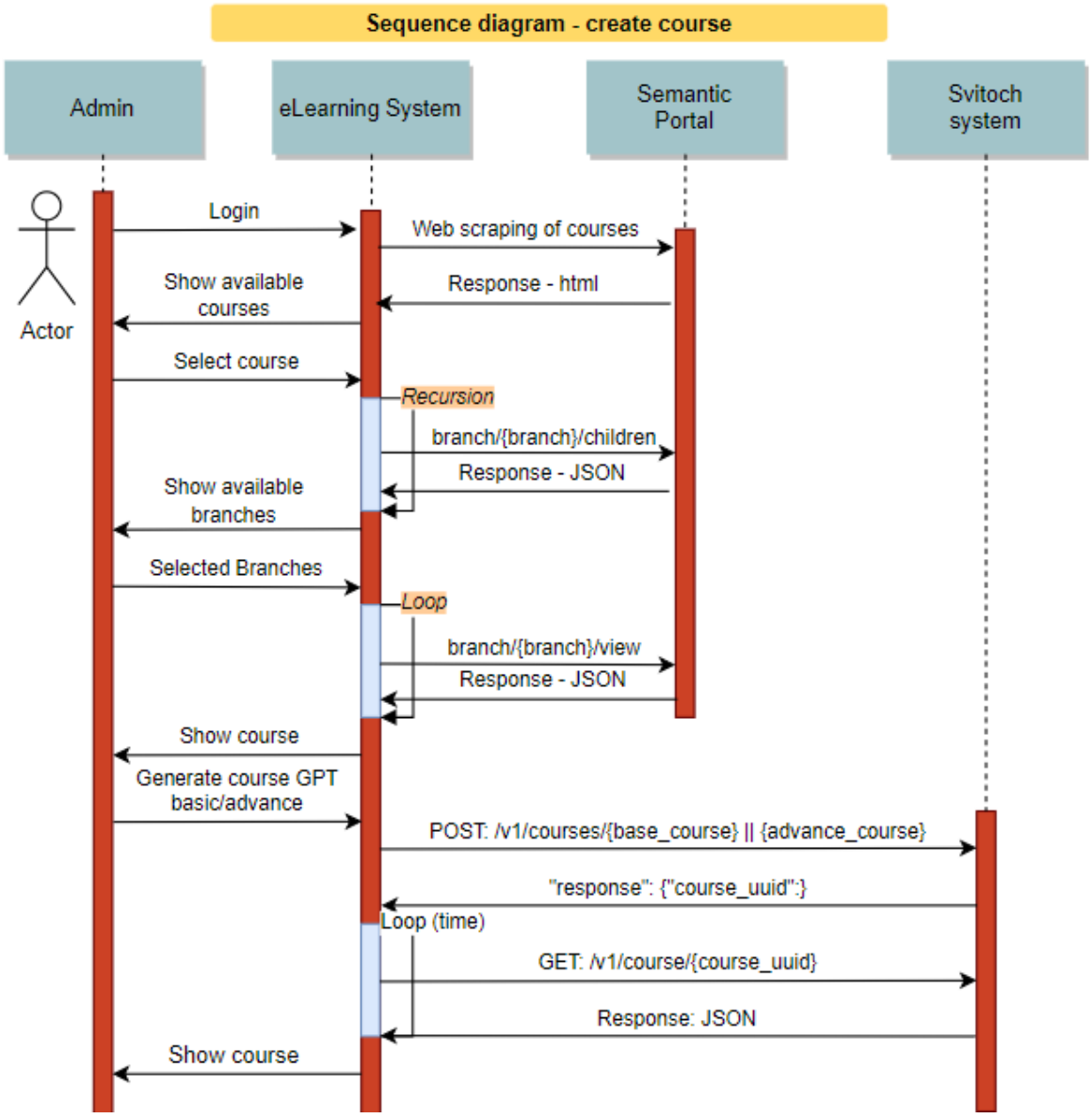


Figure 2.6 Sequence diagram of course creation workflow

Process Steps

Administrator Login: The process initiates with the Administrator logging into the eLearning System, establishing a secure session to manage course content.

Course Selection Interface: Upon successful authentication, the system presents the Administrator with a list of available courses. The Administrator selects a course to manage or update, triggering the next steps in the sequence.

Web Scraping of Course Data: The eLearning System interacts with the Semantic Portal, scraping the web for course data. The system receives an HTML response, including the structure and content of the course offerings on the Semantic Portal.

Retrieving Course Branches: With the course selected, the eLearning System performs a recursive API call to branch/{branch}/children on the Semantic Portal. This action fetches all related subtopics (children) for the selected course, with the response returned in JSON format.

Displaying Available Branches: The retrieved branches are then displayed to the Administrator, allowing the selection of specific branches or subtopics that will be included in the course content.

Branch Selection and Data Gathering: After the Administrator selects the relevant branches, the system loops through each branch, making additional API calls to branch/{branch}/view. This step collects detailed information about each branch in JSON format, including content, descriptions, and associated metadata.

Course Generation with GPT: The collected branch data is input for the course generation process. The Administrator can choose to generate a basic or advanced course using GPT (presumably a model like ChatGPT), which the Svitoch system facilitates. Depending on the course level, a POST request is sent to v1/courses/{base_course} or v1/courses/{advance_course}.

Polling and Response Handling: After initiating the course generation, the system periodically polls the server with a GET request to v1/course/{course_uuid}. This is a waiting mechanism to receive the generated course content, which, upon completion, is returned as a JSON response.

Finalization and Display of Course: The final step in the sequence is displaying the newly generated course content to the Administrator. This step is crucial as it allows for a review of the generated content, ensuring it meets the quality and educational standards before being made available to end-users.

2.7 Database Structure

SQLite is the database of choice for the eLearningPortal project. Its simple, file-based nature eliminates the need for a separate server, making it an ideal solution for managing the content within the system. SQLite's lightweight design is particularly advantageous for the eLearningPortal project, as it requires minimal setup and maintenance while providing all the necessary functionality for data storage and retrieval.

Data Extraction: Initiation begins with data extraction from the semantic portal, which systematically collects available courses and their details.

Content Generation: The Svitoch project API takes the structured data and interacts with ChatGPT to generate new course content.

Database Interaction: All generated content from the semantic portal or ChatGPT is stored in the SQLite database and ready for admin review.

Admin Interface: The admin interface is a critical component, facilitating the management and approval of courses. It provides tools for course comparison, selection, and regeneration.

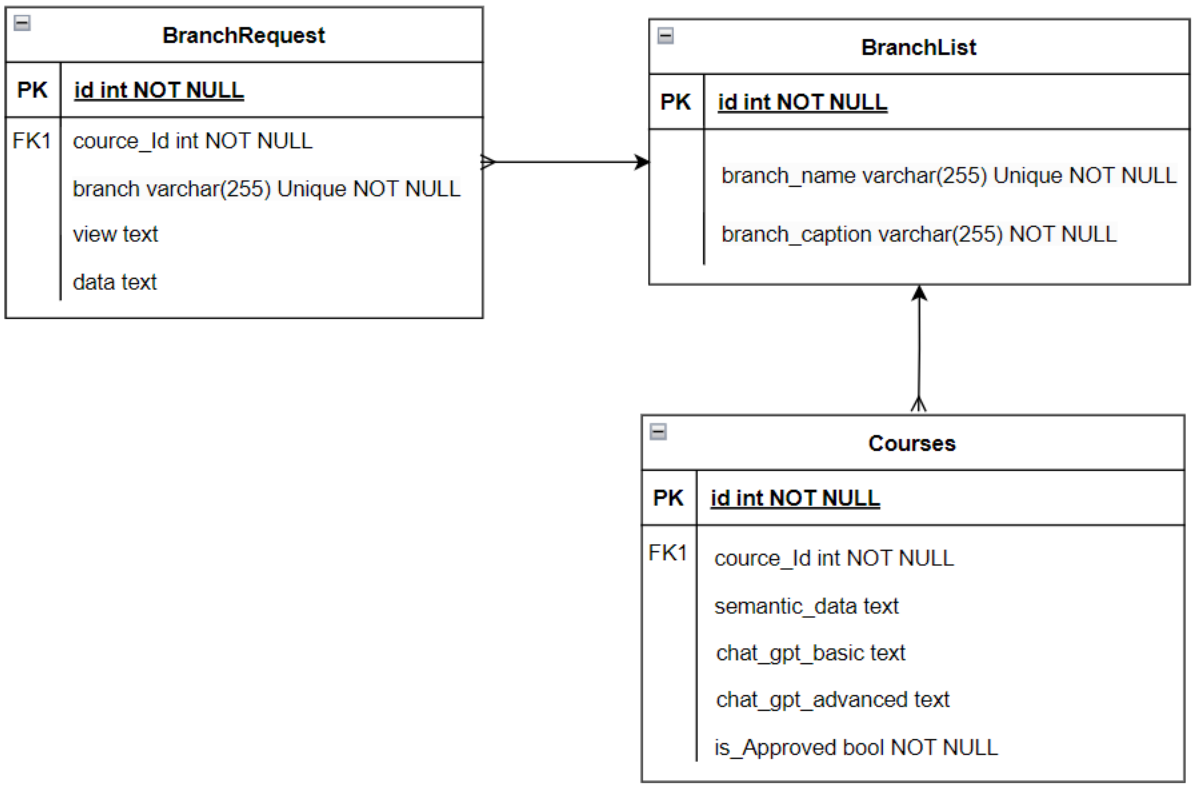


Figure 2.7 Entity relationship diagram

BranchRequest and Course models can dynamically store a wide range of structured data thanks to JSON fields. This offers great flexibility in handling diverse content types generated by the Semantic Portal and ChatGPT. The __str__ methods enhances the admin's user experience by providing clear, identifiable names for each database record.

The structure includes a system where the educational content that exists on the portal is stored in BranchRequest, in BranchList we store all data about child pages and the content itself for topics, and ultimately Course stores generated educational modules that are dynamically created by the administrator and are easily managed via the Django admin interface or custom views.

BranchRequest: A model that links to the BranchList through a foreign key, implying a relationship where multiple branch requests may correspond to a single branch list entry.

The branch field is a unique identifier for a particular topic or subject within a course.

The JSON structure data field stores metadata or descriptive content about the branch.

The view field, also a JSON object, may contain more detailed information, such as HTML views or specific content related to the branch.

BranchList: This model serves as a catalog of all branches or subjects available within the portal.

The branch_name is a unique textual identifier for each branch.

The branch_caption provides a more descriptive or user-friendly name for the branch.

Course: Represents the actual courses offered within the eLearning portal.

It is linked to BranchList, suggesting that each course is associated with a particular branch.

The model houses JSON fields for semantic_data, chat_gpt_basic, and chat_gpt_advance, each potentially storing different tiers or versions of the course content.

The is_approved boolean field indicates whether the course content has been reviewed and approved, a critical feature for maintaining content quality.

This database is ideal for development and debugging. It is very easy to use and integrates into the project, which does not require additional settings. But when the stored information increased, freezes for several seconds were noticed. Thanks to the well-thought-out Django framework, this database can be replaced in the project settings and migrations can be regenerated, which will lead to the creation of a new database. In the future, it is advisable to implement it on another more powerful database such as PostgreSQL.

3. User guide for admin

The main page of the eLearning portal showcases a clean and navigable interface with a prominent banner for American University Kyiv and quick-access buttons for courses and login. Below is a selection of course categories inviting users to explore the educational offerings.

Figure 3.1 Main page

The project repository is available at:

https://github.com/IvchenkoSanya/Capstone

Login

To gain access to advanced features on the eLearningPortal, the administrator is required to enter credentials in the 'Login' and 'Password' fields and then proceed by clicking the 'Login' button.

Login: admin
Password: 1234

3.1 Generation a new course based on Semantic Portal

Step 1: To initiate the creation of a new course based on offerings from the Semantic Portal, an administrator needs to click the 'Semantic Portal' button.

Step 2: Clicking on the 'Semantic Portal' button navigates to a new page featuring a dropdown list, allowing for the selection of a existing course and click Submit.

Step 3: Consequently, you will be presented with all the branches related to the selected course, including all the child pages available on the Semantic Portal resource. To continue click generate personal course button.

Step 4: Choose the topics that pique your interest and then click the 'Submit' button to proceed

Upon selection, the resulting page will display a course generated based on the topics chosen from the Semantic Portal.

3.2 Generation a new course manually

Click button Write Course

In the provided form, please enter details in the 'Course' and 'Topics' fields.

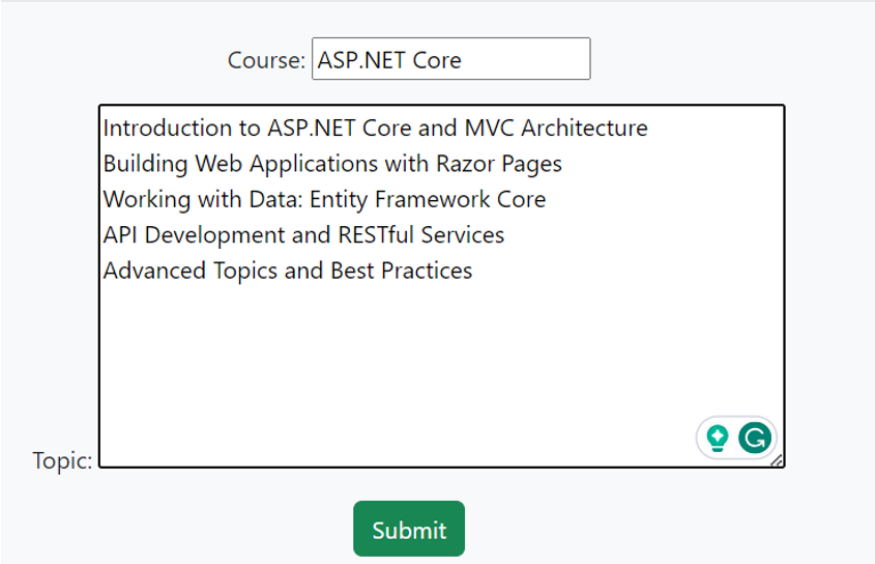


Figure 3.2 Example of filled form

3.3 Admin panel

Navigate to http://127.0.0.1:8000/admin/ in your browser. This will grant you access to extended administrative functionalities, where you can modify database information and approve courses.

In the administrative panel, navigate to "SemanticPortal" then to "Courses" and select the interesting course.

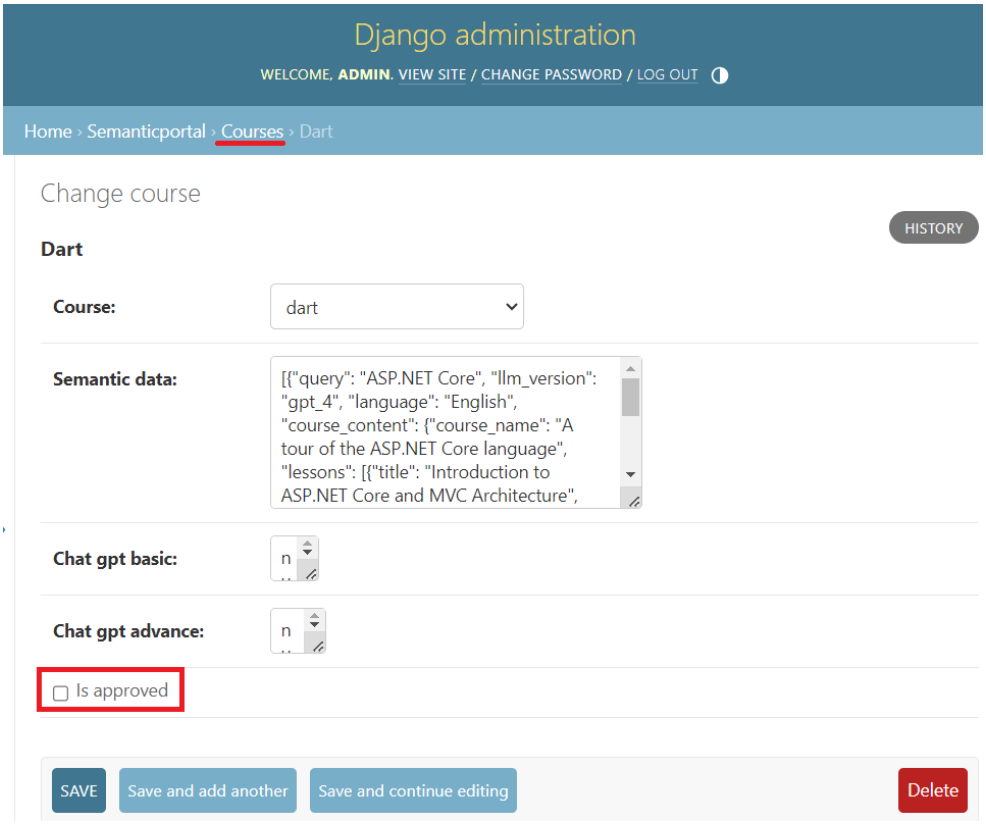
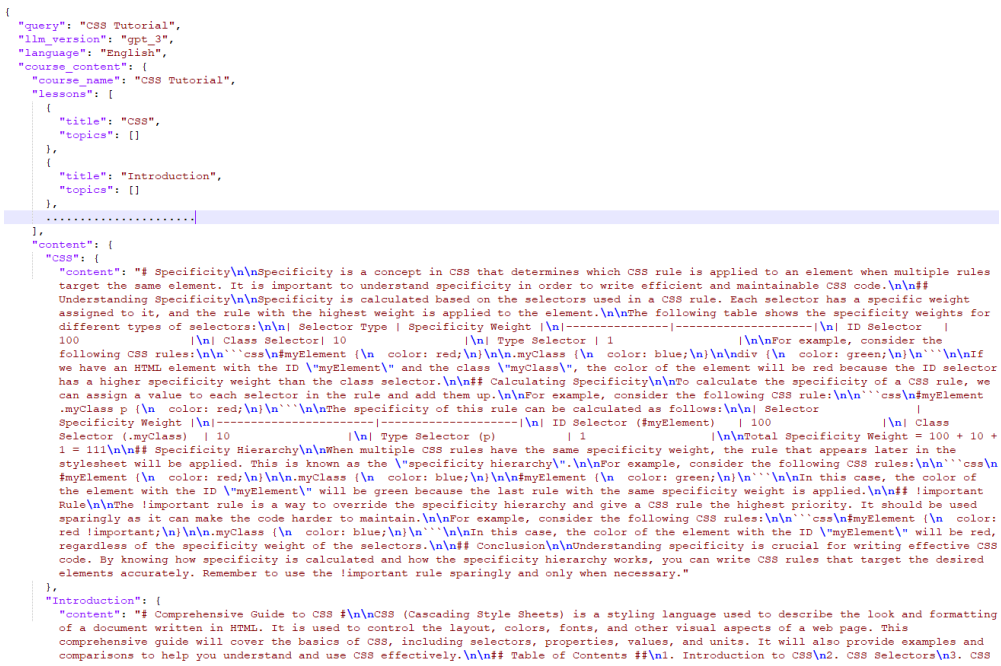


Figure 3.3 Course approval button

3.4 JSON Output Formatted in Markdown

We receive all responses in JSON format, a universally recognized data interchange format, which enhances the interoperability of our system with various external services and APIs. This format is particularly advantageous due to its lightweight nature and ease of parsing, which is crucial for efficient data handling in web applications. By utilizing JSON, we ensure that data transmission between our educational portal and the integrated LLM-based content generation systems is both seamless and reliable.

The structured nature of JSON responses also facilitates the rapid and accurate processing of data within our system. For instance, when educational content is generated and sent back from the LLM system, the JSON format allows for straightforward extraction and incorporation of this content into our portal. This efficiency not only improves the user experience by providing timely and relevant educational materials but also streamlines backend operations, enhancing the overall performance of the system. The use of JSON, therefore, is a key aspect of our system architecture, contributing significantly to the agility and effectiveness of the educational platform.



```
{
  "query": "CSS Tutorial",
  "llm_version": "gpt_3",
  "language": "English",
  "course_content": {
    "course_name": "CSS Tutorial",
    "lessons": [
      {
        "title": "CSS",
        "topics": []
      },
      {
        "title": "Introduction",
        "topics": []
      },
      ......................
    ],
    "content": {
      "CSS": {
        "content": "# Specificity\n\nSpecificity is a concept in CSS that determines which CSS rule is applied to an element when multiple rules
        target the same element. It is important to understand specificity in order to write efficient and maintainable CSS code.\n\n##
        Understanding Specificity\n\nSpecificity is calculated based on the selectors used in a CSS rule. Each selector has a specific weight
        assigned to it, and the rule with the highest weight is applied to the element.\n\nThe following table shows the specificity weights for
        different types of selectors:\n\n| Selector Type | Specificity Weight |\n|---------------|--------------------|\n| ID Selector    |
        100               |\n| Class Selector| 10                 |\n| Type Selector | 1                  |\n\nFor example, consider the
        following CSS rules:\n\n```css\n#myElement {\n  color: red;\n}\n\n.myClass {\n  color: blue;\n}\n\ndiv {\n  color: green;\n}\n```\n\nIf
        we have an HTML element with the ID \"myElement\" and the class \"myClass\", the color of the element will be red because the ID selector
        has a higher specificity weight than the class selector.\n\n## Calculating Specificity\n\nTo calculate the specificity of a CSS rule, we
        can assign a value to each selector in the rule and add them up.\n\nFor example, consider the following CSS rule:\n\n```css\n#myElement
        .myClass p {\n  color: red;\n}\n```\n\nThe specificity of this rule can be calculated as follows:\n\n| Selector            |
        Specificity Weight |\n|---------------------|--------------------|\n| ID Selector (#myElement)    | 100                |\n| Class
        Selector (.myClass)   | 10                 |\n| Type Selector (p)           | 1                  |\n\nTotal Specificity Weight = 100 + 10 +
        1 = 111\n\n## Specificity Hierarchy\n\nWhen multiple CSS rules have the same specificity weight, the rule that appears later in the
        stylesheet will be applied. This is known as the \"specificity hierarchy\".\n\nFor example, consider the following CSS rules:\n\n```css\n
        #myElement {\n  color: red;\n}\n\n.myClass {\n  color: blue;\n}\n\n#myElement {\n  color: green;\n}\n```\n\nIn this case, the color of
        the element with the ID \"myElement\" will be green because the last rule with the same specificity weight is applied.\n\n## !important
        Rule\n\nThe !important rule is a way to override the specificity hierarchy and give a CSS rule the highest priority. It should be used
        sparingly as it can make the code harder to maintain.\n\nFor example, consider the following CSS rules:\n\n```css\n#myElement {\n  color:
        red !important;\n}\n\n.myClass {\n  color: blue;\n}\n```\n\nIn this case, the color of the element with the ID \"myElement\" will be red,
        regardless of the specificity weight of the selectors.\n\n## Conclusion\n\nUnderstanding specificity is crucial for writing effective CSS
        code. By knowing how specificity is calculated and how the specificity hierarchy works, you can write CSS rules that target the desired
        elements accurately. Remember to use the !important rule sparingly and only when necessary."
      },
      "Introduction": {
        "content": "# Comprehensive Guide to CSS #\n\nCSS (Cascading Style Sheets) is a styling language used to describe the look and formatting
        of a document written in HTML. It is used to control the layout, colors, fonts, and other visual aspects of a web page. This
        comprehensive guide will cover the basics of CSS, including selectors, properties, values, and units. It will also provide examples and
        comparisons to help you understand and use CSS effectively.\n\n## Table of Contents ##\n1. Introduction to CSS\n2. CSS Selectors\n3. CSS
```


Figure 2.3.17. Example of returned JSON

Content in JSON returned as Markdown. Markdown is a simple markup language known for its plain-text formatting capabilities. It is designed to be easily convertible into a variety of output formats, but it's most commonly used to generate HTML.

Conclusion

Developing the Integrated Learning System (ILS) subsystem involved extensive research and analysis in digital education. Emphasizing the need for adaptive and personalized learning experiences, the project aimed to create a system that facilitates educational content creation and integrates advanced technologies for a more dynamic learning environment. The research underscored the importance of a user-friendly interface, efficient content management, and the ability to integrate with external educational resources.

1. The Integrated Learning System blends user-centric design with robust backend architecture, creating a system that enhances the educational experience for administrators and learners. The system's ability to integrate with LLM-based educational content generation systems and educational resources positions it as a forward-thinking solution in digital education. The project highlights technology's critical role in advancing educational methodologies and accessibility.

2. The resulting system is developed using Python and Django, reflecting best practices in modern software engineering. The system stands out for its seamless user interface, which offers administrative control over content creation and management. This intuitive and responsive interface makes it accessible across various devices and screen sizes. It simplifies administrative

tasks such as course creation, content editing, and quality control, ensuring a streamlined process for educators and administrators.

3. Several key modules constitute the core of the ILS:

User Interface Module: This module provides a web-based portal where administrators can manage educational content. It features a clear, navigable layout, allowing efficient system interaction.

Backend Services Module: The backbone of the system, this module handles business logic, data processing, and API interactions. It ensures smooth system operation and facilitates the integration of external resources.

Integration Module: This critical module connects the portal with external content generation services like the Svitoch System and educational resources from the Semantic Portal. It enables the system to offer diverse educational materials and LLM-based educational content generation.

Database Module: Utilizing an SQLite database, this module stores course metadata, user profiles, and educational content. It's designed for performance and ease of management.

4. The capabilities that enhance the system features

Content Storage Subsystem: This subsystem automatically stores the data in a structured format within the SQLite database.

Course Creation and Management Subsystem: It allows administrators to assemble course structures from available content and leverage LLM-based models for generating diverse course materials.

Quality Assurance Functionality: This feature enables administrators to review and approve AI-generated content, ensuring the material's quality and relevance. It also facilitates the comparison of traditional and LLM-based educational content generation.

User Accessibility and Interaction: The system's interface allows end-users to access and interact with course materials, contributing to an engaging learning experience.

5. This system made it possible to critically evaluate the capabilities of the LLM-based educational content generation system and existing high-quality materials. For example, the existing course content, traditionally structured and replete with examples, demonstrates an orderliness conducive to learning. This structured approach aligns well with educational standards, presenting material logically that builds understanding incrementally. Nevertheless, it's essential to acknowledge the substantial merit of the content produced by the LLM-based educational content generation system. The information it generates often exhibits a high degree of accuracy and relevance. The LLM's ability to pull from a vast data repository allows it to offer informative content rivaling traditional sources. Moreover, the continuous improvements in LLM models herald a future where

the gap in quality and organization between AI-generated content and meticulously curated courses may narrow significantly.

6. Using the developed system, it was possible to analyze and identify shortcomings that an administrator may encounter at the current iteration. In generating extensive responses, ChatGPT can sometimes include redundant information, leading to verbosity that might obscure key points rather than elucidate them. Conversely, when brevity overrides, the answers may lack the depth necessary for comprehensive understanding, particularly in responses to complex queries. ChatGPT may struggle to convey the intricacies required for deep comprehension regarding complex programming topics or advanced subject matter. Such topics often necessitate information and insight honed through years of specialized study and experience. Despite these challenges, it can assist teachers by offering alternative explanations, brainstorming ideas, or providing a starting point for lesson planning and material development. For students, it is a source of supplementary knowledge.

7. However, one must exercise discernment in leveraging LLM-based educational content generation. Complex issues requiring expert analysis still demand human expertise. The subtleties of certain subjects, the critical evaluation of sources, and the application of advanced concepts in practical scenarios require a depth of understanding beyond current LLM capabilities. Integrating LLM like ChatGPT in educational settings presents a mosaic of opportunities and challenges. It can enhance educational experiences by extending traditional learning resources. Yet, it should be complemented with expert guidance, particularly for intricate subjects requiring a profound understanding. As AI technology evolves, its role in education may expand, potentially transforming the learning and teaching landscape.

Bibliography

1. Haluza, D., & Jungwirth, D. (2023). Artificial Intelligence and ten societal megatrends: a GPT-3 case study. Preprints, 2023010474 (doi: 10.20944/preprints202301.0474.v1)

2. Mhlanga, D. (2023). Open AI in Education, the Responsible and Ethical Use of ChatGPT Towards Lifelong Learning. Education, the Responsible and Ethical Use of ChatGPT Towards Lifelong Learning (February 11, 2023)

3. Ferster, B. (2014). Teaching machines: Learning from the intersection of education and technology. JHU Press.

4. Rudolph, J., Tan, S., & Tan, S. (2023) ChatGPT: Bullshit spewer or the end of traditional assessments in higher education?. Journal of Applied Learning and Teaching, 6(1)

5. Srinivasa, K. G., Kurni, M., & Saritha, K. (2022). Harnessing the Power of AI to Education. In Learning, Teaching, and Assessment Methods for Contemporary Learners (pp. 311-342). Springer, Singapore.

6. Patil, A. S., & Abraham, A. (2010). Intelligent and Interactive Web-Based Tutoring System in Engineering Education: Reviews, Perspectives, and Development. In Computational Intelligence for Technology Enhanced Learning (pp. 79-97). Springer, Berlin, Heidelberg.

7. Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F. & Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education.

8. Rudolph, J., Tan, S., & Tan, S. (2023). ChatGPT: Bullshit spewer or the end of traditional assessments in higher education. Journal of Applied Learning and Teaching, 6(1).

9. Zhai, X. (2022). ChatGPT user experience: Implications for education. Available at SSRN 4312418

10. Mohammad Awad AlAfnan, Samira Dishari, Marina Jovic, Koba Lomidze (2023). ChatGPT as an Educational Tool: Opportunities, Challenges, and Recommendations for Communication, Business Writing, and Composition Courses. https://doi.org/10.37965/jait.2023.0184

11. Changrong Xiao, Sean Xin Xu, Kunpeng Zhang, Yufang Wang, Lei Xia (2023). Evaluating Reading Comprehension Exercises Generated by LLMs: A Showcase of ChatGPT in Education Applications.(10.18653/v1/2023.bea-1.52)

12. Sami Sarsa, Paul Denny, Arto Hellas, Juho Leinonen (2022). Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models (https://doi.org/10.1145/3501385.3543957)

13. Lund, B. D., & Wang, T. (2023). Chatting about ChatGPT: how may AI and GPT impact academia and libraries Library Hi Tech News.

14. Botha, J., Grobler, M. M., Hahn, J., & Eloff, M. (2017). A high-level comparison between the South African protection of personal information act and international data protection laws. In ICMLG2017 5th International Conference on Management Leadership and Governance (p. 57)

15. Zhai, Xiaoming, ChatGPT and AI: The Game Changer for Education (March 15, 2023). SSRN 4389098

16. Tytenko, S.V. (2019) Generation of test tasks in the system of distance learning based on the model of formalization of didactic text. Scientific news of NTUU "KPI",№1(63),p.47–57. URL: http://www.setlab.net/?view=Tytenko-test-generation . Article sent: 19.10.2023