Bridging Language Barriers in AI Education: A Smart Platform Integrating Speech-to-Text, Translation, and Real-Time Knowledge Delivery

Jae Young Woo [1] and Ill Chul Doo [2,*]

[1] Department of Computer and Electronic Systems Engineering and Department of Mathematics (Double Major), Faculty of Natural Sciences, Hankuk University of Foreign Studies, Gyeonggi, South Korea
[2] Artificial Intelligence Education, Faculty of Engineering, Hankuk University of Foreign Studies, Seoul, South Korea
Email: hufswo@gmail.com (J.Y.W.); dic@hufs.ac.kr (I.C.D.)
*Corresponding author

***Abstract*—This study introduces an Artificial Intelligence (AI)-based education platform that enables multilingual learning by integrating real-time speech recognition, Speech-to-Text (STT) translation, automated news parsing, and chatbot support functions. By utilizing Whisper-based speech recognition, Google API multilingual translation, Selenium-based news crawling, and an OpenAI chatbot, learners can access translated lecture scripts, stay informed about the latest AI developments in real time, and receive personalized learning support through natural language interaction. The platform is designed to enhance accessibility, eliminate language barriers, and integrate fragmented educational resources into a cohesive system. System evaluation demonstrated high performance, recording a Word Error Rate (WER) of 4% and a Bilingual Evaluation Understudy (BLEU) score of 85, validating the accuracy and reliability of the transcription and translation modules. A System Usability Scale (SUS) evaluation conducted with 30 participants yielded an average score of 79.5, indicating high user satisfaction across diverse age and experience groups. The modular and reproducible system architecture ensures adaptability to different languages and educational environments. This platform contributes to establishing a standard model for global AI-based education systems by supporting scalable and personalized learning. Future research will focus on expanding language support, enhancing context-aware translation accuracy, developing data-driven employment matching capabilities, and implementing effective strategies for broader service deployment. This integrated approach aims to address current limitations in AI education accessibility and foster the development of global digital talent.**

***Keywords*—multilingual education, speech recognition, Artificial Intelligence (AI) chatbot, speech to text translation technology, automation crawling and parsing, real-time learning, AI-powered platform**

I. Introduction

The emergence of OpenAI's ChatGPT model has introduced a new paradigm across various industries, leading to structural shifts in the global market. In response to these changes, the South Korean Artificial Intelligence (AI) market has been expanding rapidly. According to a domestic market outlook on artificial intelligence released by the Ministry of Trade, Industry and Energy, the size of the AI market in South Korea is projected to grow significantly by 2027, as illustrated in Fig. 1, which presents data in units of million USD [1].

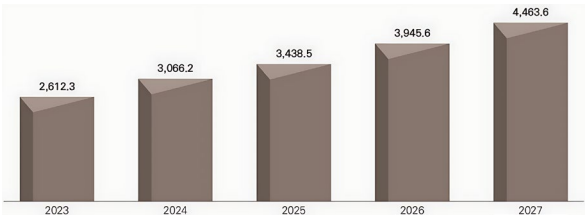

Fig. 1. Domestic AI market outlook by the Ministry of Trade, Industry and Energy, Republic of Korea.

Furthermore, a research report released by the Ministry of Employment and Labor in August 2023 predicts a shortage of 12,800 AI professionals and 18,800 cloud computing specialists over the next five years, indicating a significant labor supply-demand gap in key emerging technology fields [2]. Additionally, a study conducted by the Korea Employment Information Service in November 2024 forecasts an industrial restructuring due to digital transformation, highlighting that AI technology adoption will reduce the demand for human labor [3]. Furthermore, the global AI market is also expanding. As illustrated in Fig. 2, the market research firm Research and Markets forecasts that the global AI agent market will reach USD 33.2 billion by 2030 [4].

Fig. 2. Global AI agent market outlook.

Manuscript received April 10, 2025; revised May 6, 2025; accepted June 11, 2025; published September 12, 2025.

AI agents refer to software that autonomously utilizes tools and solves problems to achieve predefined objectives without human intervention. In other words, the expansion of the AI agent market implies a corresponding increase in the broader AI market itself.

Based on these findings, it is evident that the demand for AI professionals is increasing worldwide in response to market trends. However, according to a research report by the Institute for Information & Communications Technology Planning & Evaluation (IITP), even South Korea, which ranks among the top ten countries in AI technological capabilities, is experiencing a shortage of AI professionals relative to domestic demand [5]. One of the primary reasons for this gap is the lack of emphasis on early-stage AI and Software (SW) education. A study published by the Department of Computer Education at Busan National University of Education, titled Analysis of Overseas Software and AI Education Trends and the Necessity of Elementary School Informatics Curriculum, highlights that the amount of time allocated for software and AI education in South Korea is significantly lower than that of leading countries. The study underscores the necessity of incorporating AI-related subjects into elementary school curricula at a national level [6].

Furthermore, as mentioned in the introduction, the 2022 Information and Communications Technology (ICT) Level Survey and Technology Competitiveness Analysis Report published by the IITP indicates that the technological gap in ICT among nations continues to widen. In particular, the disparity between South Korea and the United States—the global leader in ICT—has been increasing year by year, along with other countries falling further behind [5]. Based on this data, it can be inferred that other countries with lower AI technological capabilities face similar challenges.

A review of the current status of private education providers shows that not many offer satisfactory services relative to their cost. Even the most well-known Information Technology (IT) education providers, such as Programmers, YouTube, and Inflearn, are still in a state where learners may encounter situations in which they have to study through certain lectures offered by unverified non-experts, or face limitations in the educational environment due to not supporting a wide range of programming languages. While globally renowned platforms such as Udemy and edX (jointly founded by Harvard and Massachusetts Institute of Technology (MIT)) provide high-quality courses at a reasonable cost, many of their video lectures do not support the Korean language. Particularly in the modern era, where fostering AI professionals is crucial for national competitiveness worldwide, there is a growing need for an educational system that offers courses in minority languages. Although services such as DataCamp provide partial solutions, they do not support Korean or other less commonly spoken languages. Consequently, learners who wish to access a diverse range of global AI and IT-related educational content face a significant language barrier.

Furthermore, staying updated with the latest AI advancements, which are evolving at an unprecedented pace, requires substantial effort and investment, posing additional accessibility challenges. Recent studies have highlighted the potential of AI in enhancing personalized learning and improving access to education [7, 8]. However, such systems still lack adequate support for minority languages, thereby exacerbating educational barriers for learners in linguistically marginalized regions [9]. To address this issue, AI-based educational systems designed with language independence and modularity can facilitate personalized learning across diverse educational contexts, regardless of language or location, while ensuring high reproducibility, accessibility, and scalability [10, 11]. In conclusion, there is a growing need to establish a systematic and integrated personalized educational environment for the development of AI and IT professionals. Related work is reviewed below. In recent years, research on educational chatbots has gained significant momentum, emphasizing the technical architectures and pedagogical objectives of such systems, as well as their real-time responsiveness and scalable support capabilities [8, 12]. In K-12 education, AI-based intelligent tutoring systems have demonstrated positive impacts on learning outcomes and have contributed to reducing educational disparities through AI-driven personalization [13]. The importance of AI-enabled personalized learning is also increasingly recognized in higher education, with studies systematically reviewing its impact and highlighting improvements in student engagement and academic performance [9]. A framework for learning support assistants has also been proposed to support these objectives [11]. These diverse studies collectively demonstrate the effectiveness and necessity of AI-powered educational systems.

However, most previous research has focused on individual features such as adaptive feedback or language-based tutoring. In contrast, the present study distinguishes itself by integrating multiple components—including real-time Speech-to-Text (STT) translation, AI-driven news parsing, and educational chatbots—into a unified and reproducible platform [13]. Furthermore, this system is grounded in the concept of hybrid intelligence, which combines the technical capabilities of AI with human learning needs, thereby enhancing both learner autonomy and accessibility [14]. It also directly addresses previously cited limitations in existing systems, such as insufficient linguistic diversity and restricted access to up-to-date AI knowledge [9].

Accordingly, the research objectives can be summarized as follows. This study aims to develop a customized educational system that enhances educational accessibility and improves the learner experience by utilizing AI technology in response to the increasing industrial demand for AI and IT professionals and the need for global talent development.

The complete set of survey questions, raw data summaries, and high-resolution figures used in this study are available in the supplementary materials at the following link: https://drive.google.com/drive/folders/1BtiGd35iifii0SZMU28PFu0YZH4dHD2A?usp=drive_link

II. Research Methods

The present system was designed with a modular architecture and a focus on reproducibility, drawing partial inspiration from exemplary cases of AI-based personalized education systems [11] and scalable intelligent assistant frameworks [14]. Furthermore, by incorporating a Flask-based API structure, automated AI news parsing via Selenium and APScheduler, integration of OpenAI models, and multilingual support through Whisper and GoogleTranslator AI, the system offers scalability and reproducibility that enable easy implementation across different countries and language regions. The overall framework is aligned with many intelligent assistant models proposed in previous studies [11]. However, in order to overcome the limitations and insufficient functionalities of existing approaches and to support more effective adaptive learning, this system integrates real-time STT based automatic translation and script generation, chatbot interaction, and automated parsing of up-to-date AI news, linking them as a unified set of learning support functions. Through this integration, the system actualizes a personalized and adaptive learning platform.

For all subsequent research and validation, the Republic of Korean language is set as the default language for implementation.

This study focuses on effectively integrating STT voice recognition and translation technology, automated crawling-based news parsing, and the OpenAI API to construct a multinational, specialized educational system.

The proposed system architecture consists of four functional layers, each designed with scalability, modularity, and reproducibility in mind. As illustrated in Fig. 3, the Data Layer is responsible for multilingual lecture input and AI news collection, while the Core System Framework Layer includes Flask-based API integration, a local database, and RESTful endpoints. The Intelligent Services Layer provides real-time translation, news delivery, and chatbot-supported learning assistance. Finally, the User Interface Layer integrates all components into a cohesive environment for personalized learning experiences.

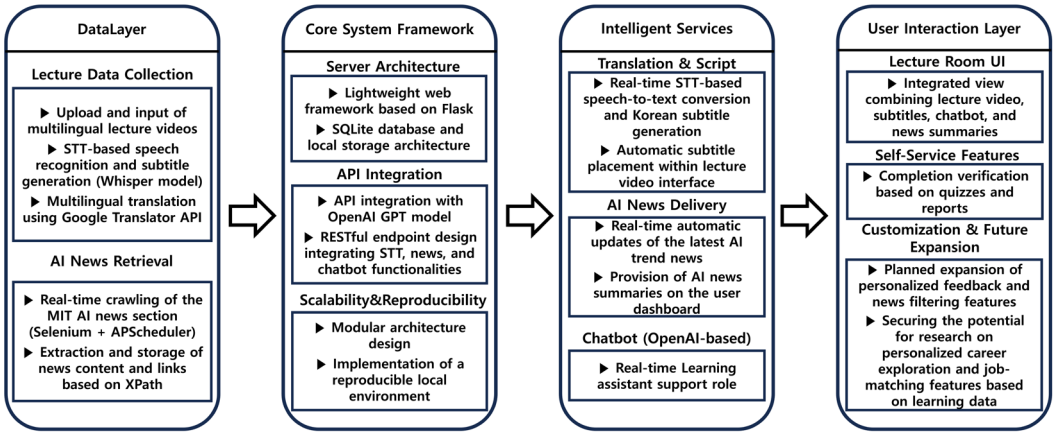

Fig. 3. Flowchart of functional system architecture.

A core functionality of the system is its ability to recognize speech in multiple languages from lecture videos, automatically convert it into text, and provide a translated script in Korean. Additionally, it features an automated data crawling function that retrieves and stores data every six hours to deliver the latest AI section news from the MIT. Furthermore, an OpenAI API-powered chatbot is implemented to assist learners during lectures.

Other supplementary features include a quiz or report-based lecture completion certification system, a coding practice environment, and a community platform.

As a result of these core functionalities, this study aims to enable learners to overcome language barriers and access a diverse range of multilingual and globally recognized lectures. Furthermore, learners can stay updated on AI and software related news without dedicating additional study time, as the latest developments are immediately available on the main page before lectures begin. Additionally, the system supports chatbot-assisted learning and assessments through quizzes and reports. By integrating previously separate services into a single platform, this study develops a customized platform for training AI professionals in South Korea by incorporating various essential features.

Subsequently, the employed technology stack and various system architectures are described, followed by a structured explanation of the core methodology and implementation results of the proposed study. The technology stacks used in each part of the system are illustrated in Fig. 4.

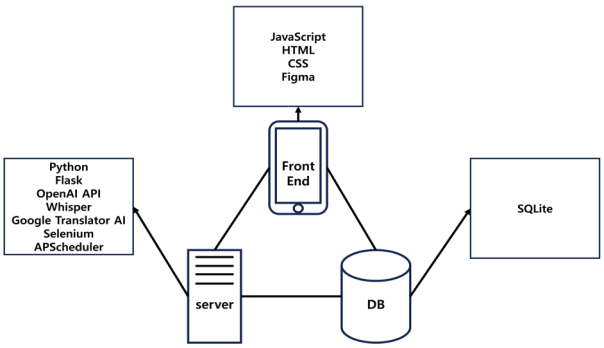

Fig. 4. Technology stack structuring.

The rationale behind the selection of each model will be discussed in detail in the functionality implementation section below.

Since a large volume of data is not required, SQLite was chosen as the database model. The main framework utilized is Flask, selected for its lightweight nature, scalability, and flexibility. Additionally, Flask is based on Python, which offers an intuitive structure and efficient request handling, making it suitable for this implementation.

A brief explanation of the design approach for each function is as follows. First, for speech recognition and translation, the Whisper STT model is employed to convert multilingual lecture videos stored in the database into text. The translated text is then processed using Google Translator AI to generate Korean language transcripts. Next, for news parsing, the Selenium model is used to crawl the desired data, which is stored in the database server. The extracted and processed information, along with relevant links, is then displayed on the main page of the site. The chatbot is developed using the OpenAI API, enabling question-answer interactions and serving as a learning assistant.

The Information Architecture (IA) design of the developed site is shown in Fig. 5, providing an overview of the system's overall structure and functional flow. The design consists of a six-depth structure, excluding unnecessary implementations beyond the core functionalities.

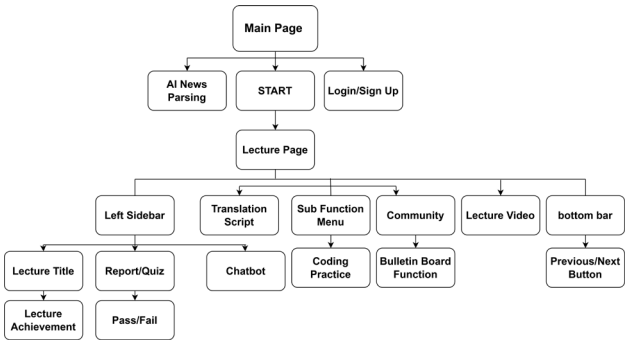

Fig. 5. Information Architecture (IA) design.

The technical design is illustrated in Fig. 6 as a simplified conceptual diagram.

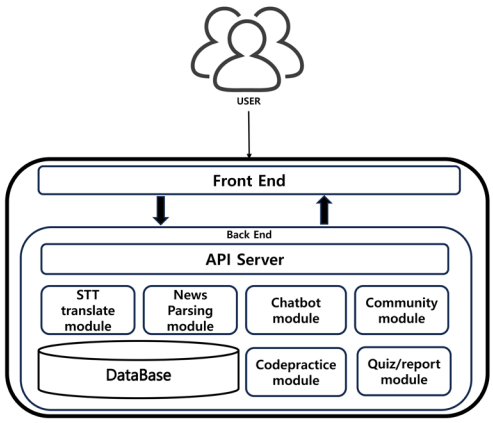

Fig. 6. System architecture diagram.

It provides an overview of how the system is deployed and how data transmission and reception occur at each stage. When a request is made from the front end, the server processes the request through the designated modules and database, then returns the corresponding values via the API.

To aid understanding, a use case diagram is presented in Fig. 7 to simulate the system's operation.

Fig. 7. Use case diagram.

By referring to the design principles and following the flow of the use case diagram, the intended system implementation can be more clearly understood. The diagram defines the requirements of the AI educational system and organizes its functions into use cases.

The key use cases depicted in the diagram are "STT Translation Script", "Automated News Parsing", and "Learning Assistant chatbot". Each use case is connected to actors, including the User and Administrator, representing interactions between the system and external entities. Additionally, sub-functions such as "Progress Tracking" and "Agreement to Terms and Conditions" are included in the diagram as part of the overall system. Furthermore, the interactions between the use cases and their associated connections are explicitly described to illustrate the system's functional relationships.

III. Implementation Results

The implementation results of each core functionality are described.

All functional design, implementation, component integration, and User Interface (UI) layout decisions were solely planned and developed by the author. The UI visualization shown in this paper was implemented in part with frontend assistance, solely for illustrative purposes. This has no bearing on the originality or academic contribution of the system architecture.

The speech recognition and translation script feature for lecture videos was developed to enable access to global lectures by overcoming language barriers. This functionality is designed by integrating Whisper, a STT model, with Google Translation AI, a multilingual translation support model, to recognize speech from stored lecture videos and convert it into text. The path of the input video file is managed within the server's storage and dynamically processed upon request. The Whisper STT model is employed to transcribe multi-lingual speech into text. The transcribed text is then translated into Korean

using Google Translator AI, and the final translated output is returned. Specifically, the server endpoint receives a lecture video processing request, converts and translates the text, and returns the resulting data in JSON format, which is dynamically displayed through the web interface for user access.

Within this framework, the algorithm and design principles were structured to form functional modules, as illustrated in Fig. 6. The final implementation follows the design shown in Fig. 8.

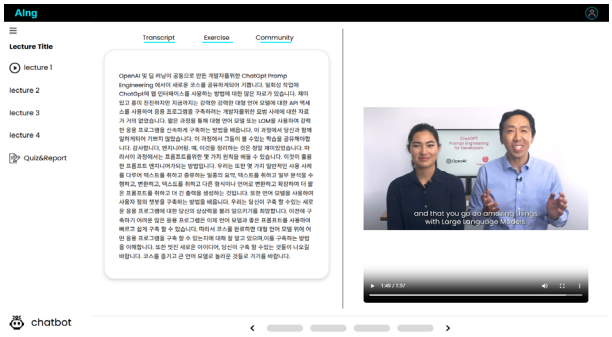

Fig. 8. Lecture room page video speech recognition and translation.

The entire process takes approximately 20 s to complete, raising a new issue concerning the need to reduce translation time. However, various derivative issues related to this will be discussed later in the results section.

The Google Translator AI model supports around 100 languages. Instead of focusing on different languages for each video, the translation experiment was conducted to assess the accuracy of translations based on pronunciation and intonation differences within the same language across speakers from non-identical linguistic regions.

Fig. 9 illustrates the translation of a deep learning lecture from the Indian Institute of Technology (IIT) [15].

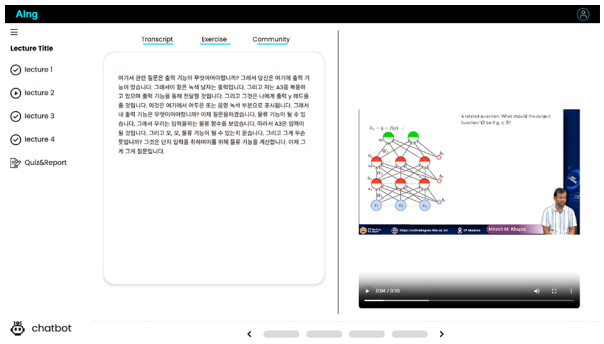

Fig. 9. Speech recognition and translation of English audio from non-English-speaking countries.

The translation was successfully completed without issues. As shown in Fig. 8, the lecture script and video were positioned on the same screen to optimize the learning experience. This functionality allows various renowned deep learning lectures from multiple countries to be effectively translated and displayed within the system [16].

Due to the lack of support for minority languages, learners in linguistically marginalized regions face significant barriers to learning and access; the STT-based automatic translation and script generation functions of this system may help alleviate these challenges [9].

Next is the automated web crawling and news parsing feature. The system is designed with the goal of enabling users to access and receive the latest AI technology trends and research materials in real time from a single platform.

Automated Crawling News Parsing (ACNP) retrieves data sources from the AI section of MIT, as illustrated in Fig. 10 [17]. This functionality is implemented using the Selenium model to crawl the relevant data, which is subsequently stored in an SQLite database. The collected data in the database server is processed to extract the required information, which is then provided via a web interface along with corresponding links and published on the main page of the site.

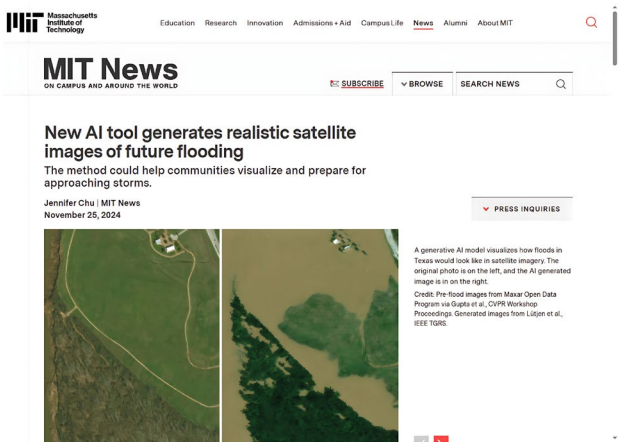

Fig. 10. MIT news AI section original page.

SQLite was selected in the early stages of system development due to its advantages in lightweight deployment, simplicity, and rapid testing capabilities. Although it is not designed for large-scale data processing, it provided sufficient performance for the implementation and validation of the system's core functionalities. As the number of users increases, future migration to more robust database management systems such as MySQL or PostgreSQL may be considered.

To provide real-time updates on rapidly evolving AI technology news, APScheduler is utilized. The data crawling process is initially executed when the server starts, and subsequently, data is automatically crawled every six hours and accumulated in the database server.

This functionality is designed using an algorithm similar to the one described earlier. The reason for adopting the crawling method instead of the simpler Really Simple Syndication (RSS) feed approach will be further discussed in the Performance Comparison and Evaluation section. The crawling process is implemented under the function name fetch latest news. At the latest ai news endpoint, the latest four crawled news articles are queried from the database and retrieved via an API using XPath. The extracted data is then customized based on the intended information structure and dynamically rendered in the client UI. As shown in Fig. 11, the extracted data is displayed at the bottom of the main page.

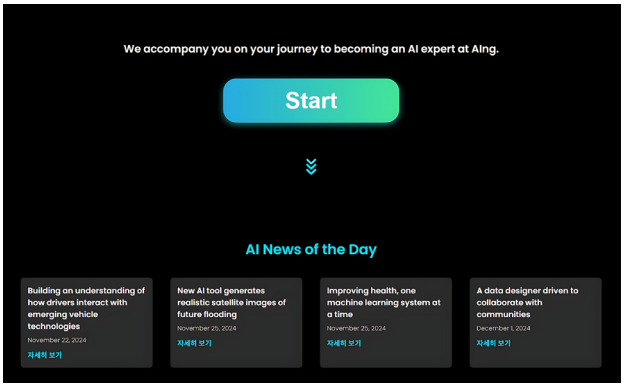

Fig. 11. ACNP function implementation screen.

The displayed information includes the news title, a brief summary, the date, and a link. Additionally, a redirection function is implemented to allow users to access the full article by clicking on the provided link.

In summary, the HTML structure of the target webpage is analyzed to parse titles, summaries, links, and publication dates. Unnecessary data is removed and the essential information is refined before being stored on the server. A portion of the parsing data processing algorithm code can be seen in Fig. 12. The real-time delivery of AI news aligns with the current trends in AI-based learning environments, which emphasize contextually relevant content and timely information updates [7].

```python
def fetch_latest_news():
    chrome_options = Options()
    chrome_options.add_argument("--headless")
    chrome_options.add_argument("--no-sandbox")
    chrome_options.add_argument("--disable-dev-shm-usage")

    service = Service(ChromeDriverManager().install())
    driver = webdriver.Chrome(service=service, options=chrome_options)

    url = "https://news.mit.edu/topic/artificial-intelligence2"
    driver.get(url)

    try:
        WebDriverWait(driver, 60).until(
            EC.presence_of_all_elements_located((By.XPATH, "//*[@id='block-mit-content']/div/div/div/div[2]/div[1]/article/div[2]/h3"))
        )

        news_data = []
        articles = driver.find_elements(By.XPATH, "//*[@id='block-mit-content']/div/div/div/div[2]/div/article")[:4]

        for article in articles:
            try:
                title = article.find_element(By.XPATH, "./div[2]/h3").text.strip()

                # Modification to resolve the URL duplication issue
                link = article.find_element(By.XPATH, "./div[2]/h3/a").get_attribute("href")
```

Fig. 12. Partial parsing code.

Finally, there is the learning assistant chatbot feature. The learning assistant chatbot module is designed to help users resolve questions in real-time during lectures. It is based on OpenAI's GPT model and provides a multi-purpose learning support service, including coding-related inquiries, lecture summarization requests, and conceptual explanations. Users input questions in natural language, and the chatbot analyzes them to provide appropriate responses, maximizing learning efficiency.

The chatbot functionality of this system is implemented based on the OpenAI GPT model and is designed with a RESTful architecture focused on question-and-answer interactions. This approach prioritizes implementation efficiency and reproducibility rather than large-scale model training. Due to the significant resource requirements and maintenance overhead associated with training Natural Language Processing (NLP) models directly, the system adopts an API integration-based method to realize its core features. Future research may enhance performance through input preprocessing, caching mechanisms, and domain-specific fine-tuning.

From an architectural perspective, user queries are transmitted to the server via RESTful API in JSON format and then relayed to the OpenAI GPT model for response generation. The OpenAI GPT-3.5 or GPT-4 model analyzes the user's input and generates a response. The generated response is returned in JSON format and displayed in the client UI, allowing users to receive real-time answers. To ensure relevant responses, a predefined "system role" is set within the OpenAI model, controlling the chatbot's context to focus on learning-related queries. The ChatCompletion API provided by OpenAI is used to handle conversations with learners.

The question processing algorithm follows these steps. The user inputs a question in natural language, which is sent to the OpenAI endpoint in JSON format. The GPT model analyzes the question and generates a response. The generated response is returned to the client and displayed on the screen. No separate NLP module is implemented, as the OpenAI API is used directly. For maintaining conversation context, the algorithm continuously stores previous interactions to provide consistent responses to follow-up questions. However, unlike standard GPT models, a visible chat history UI is not implemented, as the chatbot is specifically placed within lecture subtopics to support topic-based learning. Since the algorithm utilizes the OpenAI API server, no additional database or modules were created, simplifying maintenance. While this approach offers ease of use, it also presents limitations, such as difficulties in model optimization and customization.

For convenience, the chatbot is placed within the lecture room page, as shown in Fig. 13.

When appropriately integrated into the system, such educational chatbot functions have been shown to effectively enhance learner engagement, provide scalable support, and improve personalized learning experiences [8, 12].

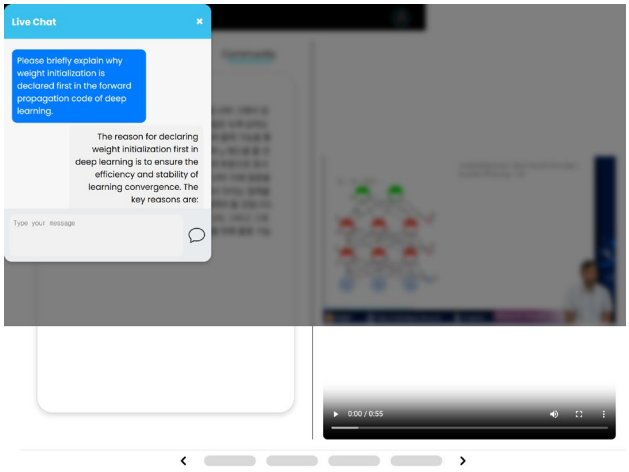

Fig. 13. Learning assistant OpenAI-based chatbot.

IV. Results Analysis

A. User Survey Results Analysis

To evaluate usability, functional satisfaction, and statistical reliability, a standardized survey based on the 10-item System Usability Scale (SUS) was designed and administered. The evaluation process involved showing participants a detailed simulation video and functional explanation, followed by a survey. The questionnaire was composed using a 5 point Likert scale and completed by a total of 30 participants. Additional demographic questions were included to collect information on age group, occupation, and level of AI/coding experience. The final SUS scores were calculated according to the standard scoring procedure. All responses were collected anonymously through Google Forms, with no personally identifiable information recorded. The raw responses are not publicly disclosed due to ethical considerations; however, summarized results and visualized data are presented within this paper. Participants were categorized by age group, occupation, nationality, and group size, as shown in Table I.

TABLE I. Characteristics of Survey Respondents

Number	Age	Occupation	Nationality	Personnel
1	20	University Student		8
2	20	OfficeWorker		3
3	30	OfficeWorker		4
4	30	Government Employee		1
5	30	Freelancer		1
6	30	Businessperson	Republic of Korea	1
7	40	OfficeWorker		2
8	50	OfficeWorker		4
9	50	Freelancer		1
10	50	Government Employee		1
11	50	Businessperson		4

A usability evaluation was conducted using the SUS with 30 participants. As shown in Table II, the results yielded a mean SUS score of 79.5 and a standard deviation of 11.11, indicating a high level of usability and consistent responses among participants. According to the adjective rating scale proposed by Bangor *et al*. [18], this score falls within the "Good" usability category. This suggests a high degree of user satisfaction with the system's interface and functionality, as shown in Fig. 14. The full set of survey items based on the SUS can be found in the supplementary materials at the following link: https://drive.google.com/drive/folders/1BtiGd35iifii0SZMU28PFu0YZH4dHD2A?usp=drive_link.

TABLE II. System Usability Scale (SUS) Score Summary

Evaluation Category	Mean SUS Score	Std. Deviation	Interpretation	Number of Respondents
Overall Usability	79.5	11.110,727,96	Good	30

Fig. 14 presents the distribution of SUS scores from 30 respondents. Each bar represents an individual participant's SUS score, and the dashed line indicates the mean score of 79.5.

Most scores fall within or above the "Good" usability category as defined by Bangor *et al*. [18], reflecting a generally high level of usability and consistent user satisfaction.

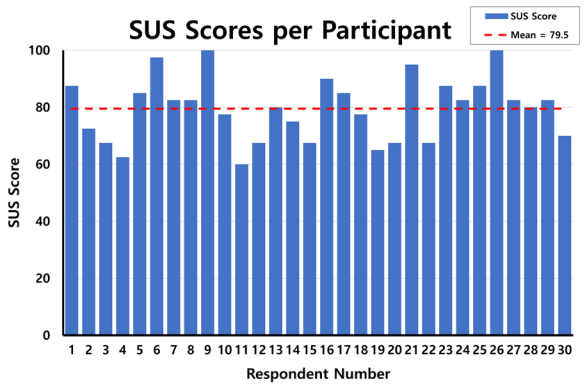

Fig. 14. Visualization of System Usability Scale (SUS) results.

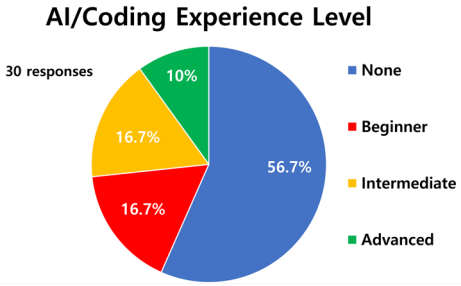

Fig. 15. Graph of AI and coding experience levels.

Fig. 15 illustrates the distribution of AI and coding experience levels among survey respondents. These supplementary questions were included to enhance the reliability of the survey, and the results indicate that the

system was considered useful by both novice and experienced users alike.

According to the user survey, the majority of participants responded very positively, suggesting that the implemented system is highly beneficial and supportive for individuals seeking to learn artificial intelligence. In addition, a separate question regarding improvement requests was conducted, and feedback related to UI/User Experience (UX) optimization, timestamp integration, and other feature enhancements will be considered in future discussions and research directions.

B. Performance Comparison and Evaluation

1) STT speech recognition and translation model performance analysis

In terms of STT speech recognition accuracy, the Whisper-based STT model is evaluated using an Automatic Speech Recognition (ASR) assessment tool. The Speech Recognition Scoring Toolkit (SCTK), specifically the Sclite command, is used to measure the Word Error Rate (WER). Additionally, the results are further analyzed using the Jiwer package. To evaluate the accuracy of STT, the WER is mathematically expressed as shown in Eq. (1).

$$\text{WER} = \frac{S + D + \text{I}}{N} \tag{1}$$

S represents the number of substituted words, *D* represents the number of deleted words, *I* represent the number of inserted words, and N represents the total number of words in the reference transcript. As a result, the WER achieved an average of 4%, indicating high accuracy. Additional experiments using ESPnet confirmed that the Whisper model effectively handles technical terms and variations in accents. Now, from an efficiency perspective, the processing time is proportional to the audio length. With the Whisper STT model, processing a 1-minute audio file takes approximately 15–20 s. The data is returned in JSON format, enhancing communication efficiency between the server and the web client.

Next, the accuracy and quality of the translation are analyzed. The Bilingual Evaluation Understudy (BLEU) score is a metric used to evaluate the performance of translation models. *BP* represents a penalty for sentence length, W_n denotes the weight of each *n*-gram, and P_n refers to precision. The translation accuracy is calculated using Eq. (2) as follows:

$$\text{BLEU} = BP \times \exp\left(\sum_{n-1}^{N} W_n \log P_n\right) \tag{2}$$

The evaluation results show that the BLEU score achieved 85 points (for text containing technical terms), while the Translation Error Rate (TER) remained below 15%, demonstrating high accuracy and the capability of providing high-quality multilingual translations. Additionally, the METEOR score was 87 points, indicating a positive assessment in terms of contextual relevance.

The following section presents a comparison between different speech translation models. The advantages and disadvantages of translation time across different speech translation models are presented in Table III.

TABLE III. COMPARISON OF SPEECH TRANSLATION MODELS

Criterion	Google Translate API	DeepL API
Speed	High (within 1 s)	Medium (1.5 s)
Language Support	100+	30+
Context Processing	Limited context awareness	Excellent context awareness
Cost	Low	Relatively high

In the key languages used in this project, there is minimal difference in contextual interpretation. For services where user response time is the most critical factor, the Google Translate API was chosen due to it fast processing speed. Considering the possibility of expanding the service beyond South Korea, a model that supports a wide range of languages is advantageous. Additionally, compared to DeepL, Google Translate API is more intuitive and can be easily integrated using the googletrans library in Python. It is also well compatible with frameworks like Flask, making maintenance more efficient.

2) News parsing performance analysis

The advantages and disadvantages of using RSS feeds for data parsing versus automatically storing parsed data in a database through scheduled web crawling are presented in Table IV.

A summary of the key differences is shown in Table IV. If the primary goal is simply to display the latest AI news quickly, the RSS feed method is preferable. However, if the goal is to process and store news updates for customized functionality, the crawling method is more useful.

For this reason, the crawling approach was chosen. Additionally, since the system requires continuous crawling to provide the most up-to-date information, APScheduler is utilized to automate crawling at scheduled time intervals.

TABLE IV. COMPARISON OF RSS AND WEB CRAWLING METHODS

Criterion	RSS Feed Method	Crawling Method
Real-time Capability	High	Relatively low (Updated at preset intervals)
Implementation Complexity	Low	High (Requires understanding of HTML structure and maintenance of crawling code)
Data Control	Low (Limited to provided data)	High (Data can be processed in the desired format)
Server Resource Consumption	Low (Consumes minimal resources)	High (Periodic crawling may cause server load)

The subsequent section provides a comparison of the models. For this functionality, Selenium is more suitable. Table V presents a comparison of advantages and disadvantages between Selenium and BeautifulSoup, the two most widely used web scraping models.

TABLE V. COMPARISON OF SELENIUM AND BEAUTIFULSOUP

Criterion	Selenium	BeautifulSoup
Handling Dynamic Content	Supports JavaScript rendering, enables dynamic page crawling	Does not support JavaScript rendering, only processes static HTML
Speed	Slow due to browser execution	Fast as it only performs HTML parsing
Installation & Ease of Use	Requires browser driver installation, complex setup	Lightweight library, easy to install and use
Resource Consumption	High CPU/memory usage due to browser execution	Low resource consumption due to lightweight processing
Interactivity	Can simulate user actions such as button clicks and scrolling	Only supports static HTML analysis
Use Cases	Crawling dynamic web pages	Parsing static HTML page data

Selenium is capable of handling dynamic content and simulating user interactions, making it effective for retrieving JavaScript-generated news data from MIT Technology Review. In contrast, BeautifulSoup is optimized for static HTML and has limitations when crawling dynamic websites like MIT Technology Review. For speed and resource optimization, one possible approach is caching the crawled data from Selenium. Additionally, combining both models could be an effective strategy when necessary.

The following proceed to analyzes to the accuracy. To ensure that duplicate data is removed and that essential information such as news titles, links, and dates is accurately extracted, print statements are used during server execution to automatically check the output results. This allows for an intuitive assessment of whether crawling was successfully performed, as demonstrated in Fig. 16.

```
            # Modification to resolve the URL duplication issue
            link = article.find_element(By.XPATH, "./div[2]/h3/a").get_attribute("href")

            # Check and handle whether the href is an absolute URL
            if not link.startswith("http"):
                link = "https://news.mit.edu" + link

            summary = article.find_element(By.XPATH, "./div[2]/p[1]/span").text.strip()
            date = article.find_element(By.XPATH, "./div[2]/p[2]").text.strip()

            news_data.append((title, link, date))
        except Exception as e:
            print("Error occurred during crawling:", e)
except TimeoutException:
    print("Page load timeout - Failed to find the element.")
    news_data = []
finally:
    driver.quit()

# Save to database
conn = sqlite3.connect(DATABASE)
cursor = conn.cursor()
cursor.execute("DELETE FROM news")
cursor.executemany("INSERT INTO news (title, url, date) VALUES (?, ?, ?)", news_data)
conn.commit()
conn.close()

print("Crawled news data:", news_data)
```

Fig. 16. Parsing data accuracy analysis code.

The performance accuracy of the automated web crawling system is calculated using Eq. (3).

$$\text{Success Rate} = \frac{\text{Successful Crawls}}{\text{Total crawls}} \times 100\% \quad (3)$$

The evaluation results show that the crawling success rate was measured at approximately 95%, with the remaining 5% attributed to network latency and page load errors.

Lastly, an analysis of efficiency is conducted. The average crawling time was measured at approximately 1 min and 30 s, and the database updates are executed immediately after crawling. The API response time for data retrieval is approximately 200 ms on average, allowing for fast data rendering on the client side.

3) Chatbot response quality evaluation

To analyze the chatbot's response speed, the response time can be mathematically expressed using Eq. (4).

$$T_{response} = T_{processing} + T_{network} + T_{model} \quad (4)$$

Here, Tprocessing represents the server-side data processing time, Tnetwork denotes the network transmission latency, and Tmodel refers to the response generation time of the OpenAI GPT model. However, since the chatbot functionality was implemented using OpenAPI without additional preprocessing, no accuracy or efficiency analysis was conducted.

4) Execution time analysis of core function algorithms (Big-O)

a) STT speech recognition and translation script function

The execution time of the Whisper model increases proportionally to the length of the audio (n), resulting in a time complexity of $O(n)$. Similarly, the Google Translator API increases translation time proportionally to the length of the text (m), resulting in a time complexity of $O(m)$. Since the STT conversion and translation tasks are executed sequentially, the total execution time is expressed in Eq. (5).

$$O(n + m) \quad (5)$$

b) Automated crawling for latest news parsing

The execution time of Selenium is influenced by the number of page elements to be crawled (p) and the JavaScript loading time (t), resulting in a time complexity of $O(p + t)$.

The data storage time increases proportionally to the data size (d), leading to a time complexity of $O(d)$.

Thus, the total execution time is primarily constrained by the page load and crawling stages, which act as the main bottlenecks. The overall execution time is expressed in Eq. (6).

$$O(p + t + d) \quad (6)$$

c) Chatbot

The execution time for calling the OpenAI GPT model is determined by the size of the request data (q) and the computational complexity of the model's response generation (c), resulting in a time complexity of $O(q + c)$. Here, (c) is proportional to the number of tokens processed internally by the model. Since the total execution time is

influenced by both the GPT call and the network response time, it is expressed in Eq. (7).

$$O(q + c) \tag{7}$$

5) System response time and load evaluation

The system response times for the core functionalities are as follows. The STT recognition and translation script function requires approximately 15 to 30 s to complete text conversion and translation, after which the result is returned in JSON format. The AI news crawling feature, measured based on the latest-ai-news API that returns data to the client, demonstrates a response time of less than 200 ms. For the GPT-based chatbot, the average time required to receive a JSON response from the OpenAI API is typically 2 to 3 s.

Meanwhile, the primary objective of this study is not the development of a commercial web service, but rather the mitigation of language barriers in AI education and the reproducible design and implementation of an academic learning system. Accordingly, the system was not intended for external deployment or for service-level distribution through cloud environments. All experiments were conducted within a local development environment under limited research resources. As a result, load testing under the assumption of concurrent access by multiple users, as well as scalability evaluations, were not included within the scope of this study. To compensate for these limitations, a simulation test was conducted in the local development environment by sending ten repeated requests to each of the main API endpoints namely STT conversion, AI news crawling, and chatbot queries and measuring the average response time. As a result, and as shown in Table VI, the STT conversion function recorded an average response time of 0.007 s, while the AI news crawling function showed an average of 1.523 s. These values were measured in a simplified or execution-skipped local testing environment, and in a full deployment scenario, actual response times are expected to range from 2 to 5 s, depending on input length and system configuration.

TABLE VI. RESPONSE TIMES FROM 10 REPEATED API CALLS

API Function	Average Response Time (s)	Notes
STT Conversion	0.007	Local test, actual execution takes 20–30 s
AI News Crawling	1.523	Based on latest-ai-news API
Chatbot Query Response	-	Not measured due to API key expiration

In addition, due to the expiration of the OpenAI API key during the experiment period, a precise quantitative performance evaluation of the chatbot functionality could not be conducted. However, during the implementation phase, it was confirmed that the basic response functionality operated normally, and a full performance test will be conducted once the API key is restored.

The importance of evaluating system scalability and user-based load performance is fully recognized. Future work will involve deploying the platform in a cloud environment and extending this study through follow-up research that includes scalability testing and performance validation based on real user interactions.

V. DISCUSSION

All core functionalities have been successfully developed, from initial planning to implementation, and have undergone Quality Assurance (QA) testing. Additionally, a user evaluation survey was conducted to collect user feedback. Based on this, the goal is to derive analytical results and identify areas for improvement.

According to the analysis, the system's provision of personalized learning experiences and real-time feedback is consistent with previous research findings that indicate AI can enhance learner engagement and academic performance [9, 10]. Furthermore, by integrating multilingual NLP functionality—such as translation and script generation based on the Google Translation API—this study aims to address the issue of global accessibility in AI education, thereby complementing existing research findings [7, 13].

The STT speech recognition and translation script function implemented in this study, which integrates the Whisper model with the Google Translator API, demonstrated high accuracy and efficiency. The speech recognition accuracy, measured by the WER, was approximately 4%, while the translation quality, assessed using the BLEU score, achieved 85 points. The system successfully reduced STT translation and script delivery time, allowing real-time transcription and translation of lecture content, thereby enabling learners to access global lectures without language barriers. Several optimization strategies, including model tuning, pre-translation caching, and the use of a local translation model, were considered to further reduce translation time. However, challenges such as contextual loss, cost issues, and API latency were encountered. To enhance both performance optimization and translation quality, further research is needed to maintain a balance between quality and speed, including the integration and training of custom translation models based on NLP techniques.

The Selenium-based crawling system achieved a 95% success rate in extracting data from the MIT Technology Review AI section and periodically stored the latest AI news in the database using APScheduler. This implementation allowed users to stay updated on AI research and technology trends without the need for additional searches. However, delays in loading dynamic web pages and server load issues caused by periodic crawling need to be addressed through efficient resource management and optimization of the crawling frequency.

The OpenAI GPT-based learning assistant chatbot is expected to deliver fast response times of approximately 2–3 s. It enables personalized learning material recommendations by analyzing user query data, and maintenance is simplified by relying on OpenAI's API, eliminating the need for separate model performance management. Learners could resolve their inquiries without additional costs, ensuring both cost efficiency and continuity within a single platform. To further enhance

response quality and personalized learning experiences, it is necessary to implement question analysis and classification features, as well as support for multimodal input.

Table VII summarizes the functional comparison between the proposed system and major online education platforms. All platforms possess strengths in certain functionalities.

TABLE VII. FUNCTIONAL COMPARISON WITH SIMILAR PLATFORMS

Features	Proposed System	Udemy	Inflearn	Edx	Data Camp
Real-time multilingual STT + translation	○	△	△	△	△
Automated AI news crawling, summarization	○	x	x	△	△
Real-time Q&A based on educational chatbot	○	△	x	△	○

However, each platform provides individual features to a limited extent or implements them in different ways, and no integrated learning support system has been identified.

This study differentiates itself in terms of functional integration and accessibility by organically connecting these features within a single platform.

In summary, compared to existing platforms, this system is distinguished by its integration of multilingual STT translation, AI news parsing, and chatbot-based learning support into a single cohesive workflow. This approach aligns with the recently proposed concept of hybrid intelligence [16] and also seeks to address the accessibility issues emphasized in research on AI-based tutoring and personalized learning [12, 17].

VI. CONCLUSION

This system distinguishes itself from existing platforms by integrating AI technology into education and enhancing learning accessibility. It establishes a customized educational system for training multinational AI professionals while also proposing future research directions that allow for further platform development and feature expansion.

The introduction of AI technology into education extends traditional monolingual learning environments into multilingual ones, thereby eliminating language barriers. In terms of improving learning accessibility, the system optimizes the real-time translation of lecture content and access to the latest technological resources, enabling learners to engage with global educational materials efficiently. Furthermore, the system provides a foundation for continuous exposure to the latest technological advancements, reinforcing global competitiveness and ensuring learning continuity through real-time chatbot interactions. By supporting a global learning community, this study aims to contribute to educational technology innovation and social progress.

From the perspective of limitations and future research, the accuracy of speech recognition may vary depending on intonation and pronunciation, even within the same language of a single country. Translation quality requires further improvement in handling domain-specific terminology, context sensitivity, abstract and metaphorical expressions, high-level language, and black humor. In addition, challenges remain in addressing API usage costs and latency issues during web crawling and chatbot responses.

To further elaborate on future research directions, it is essential to prioritize investigations into the challenges previously mentioned regarding the STT-based speech translation functionality. Research is needed on optimizing the Whisper model to handle variations in intonation and pronunciation, as well as addressing issues related to increased memory usage and processing speed in long lecture videos. One potential approach involves using the 'base' configuration of the Whisper model to balance performance and execution speed, while implementing data segmentation techniques that divide and process large video files in parts and subsequently merge the results, thereby reducing overall memory consumption.

The automated web crawling and news parsing functionality should also be considered as a key focus in future research. Issues such as potential crawling failures caused by delays in dynamic web page loading and the invalidation of XPath paths due to changes in external page structures require further investigation. To address the delay problem, Selenium's WebDriverWait can be utilized to ensure that target elements are fully loaded before parsing. Additionally, to maintain XPath validity, a monitoring tool could be integrated to detect HTML structure changes and automatically update XPath configurations as needed.

Finally, the future development of the chatbot functionality is considered. Research is required to address OpenAI API response latency issues and ensure accuracy in handling complex queries. Additionally, cost management is a crucial factor. For cost optimization, a business model could be established to support API costs. Alternatively, user input preprocessing and caching of redundant queries could reduce unnecessary API calls. To minimize response latency, API request responses could be processed asynchronously, improving the UX. Furthermore, response quality could be enhanced by setting contextual parameters via system messages, introducing additional filtering, and adding preprocessing stages before delivering chatbot responses.

Beyond the current scope, additional functionalities could be envisioned and investigated in future studies. Future research may explore the following functionalities. Real-time lecture video translation and script generation, service performance enhancement through automated feedback and quantitative data analysis based on actual user experiences, personalized news filtering and feed delivery according to user interests, and visualization of learning data are among the future directions. Furthermore, as the system was developed with a focus on integrated functionality, security design was not included in the current implementation. However, future work will expand on security aspects such as user authentication,

encryption of sensitive information, and API security handling.

In conclusion, this study demonstrates that it possesses reproducibility and scalability through various technology based approaches. Furthermore, the system was designed and implemented to integrate multilingual AI educational components, with additional features such as automated parsing of the latest AI news. Through this, the study aims to contribute academically to the field of AI education by enhancing accessibility and expanding the current discourse. Moreover, the quality of the research can be further improved through more concrete future proposals, such as personalized AI- and IT-related job matching, global employment information, job linkage and expert development based on learning data, global career specialization, NLP customization, support for minority languages, and identifying individual strengths and talents to cultivate global experts in specialized fields.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

JYW designed and implemented the system, conducted the experiments, and wrote the manuscript; ICD supervised the research, provided critical feedback, and edited the final manuscript; all authors had approved the final version.

FUNDING

This work was supported by Hankuk University of Foreign Studies Research Fund (of 2025).

ACKNOWLEDGMENT

The authors gratefully acknowledge everyone who contributed to the development of this paper.

REFERENCES

[1] Ministry of Trade, Industry and Energy (MOTIE). (October 2023). Standardization in industries utilizing artificial intelligence. *Tongsangnews*. [Online]. Available: https://tongsangnews.kr/webzine/1552310/special_3.html

[2] Republic of Korea Ministry of Employment and Labor. (August 2023). Report on AI workforce shortage of 12,800 and cloud workforce shortage of 18,800 by 2027. *moel.go.* [Online]. Available: https://www.moel.go.kr/news/enews/report/enewsView.do?news_seq=15458.

[3] Republic of Korea Employment Information Service. (November 2024). Industry and employment structure restructuring outlook due to digital transformation, including Artificial Intelligence (AI). *kcgkorea*. [Online]. Available: https://kcg.korea.kr/briefing/pressReleaseView.do?newsId=156659968&pWise=sub&pWiseSub=C1

[4] SK Inc, C&C. (January 2025). AI agents gaining industry attention: Use cases and future prospects. *skcc*. [Online]. Available: https://www.skcc.co.kr/insight/trend/3094

[5] Institute for Information & Communications Technology Planning & Evaluation (IITP). (March 2024). 2022 ICT technology level survey and technology competitiveness analysis, 2023 study. Republic of Korea: IITP, 2022, pp. 16–21. [Online]. Available: https://www.iitp.kr/kr/1/knowledge/openReference.it#

[6] E. S. Song and H. K. Lim, "Analysis of overseas software and AI education trends and the necessity of elementary school informatics curriculum," *J. Korean Assoc Inf. Educ.*, vol. 25, no. 2, pp. 301–308, 2021.

[7] S. Wang, F. Wang, Z. Zhu *et al.*, "Artificial intelligence in education: A systematic literature review," *Expert Systems with Applications*, vol. 252, 124167, 2024

[8] C. W. Okonkwo and A. Ade-Ibijola, "Chatbots applications in education: A systematic review," *Computers and Education: Artificial Intelligence*, vol. 2, 100033, 2021.

[9] C. Merino-Campos, "The impact of artificial intelligence on personalized learning in higher education: A systematic review," *Trends in Higher Education*, vol. 4, no. 2, 17, 2025.

[10] C.-C. Lin, A. Y. Q. Huang, and O. H. T. Lu, "Artificial intelligence in intelligent tutoring systems: Toward sustainable education—A systematic review," *Smart Learning Environments*, vol. 10, no. 41, 2023.

[11] R. Sajja, Y. Sermet, M. Cikmaz *et al.*, "Artificial intelligence-enabled intelligent assistant for personalized and adaptive learning in higher education," *Information*, vol. 15, no. 10, 596, 2024.

[12] T. Debets, S. K. Banihashem, and D. J. Brinke, "Chatbots in education: A systematic review of objectives, underlying technology and theory, evaluation criteria, and impacts," *Computers & Education*, vol. 234, 105323, 2025.

[13] A. Létourneau, M. D. Martineau, and P. M. Léger, "A systematic review of AI-driven intelligent tutoring systems in K-12 education," *npj Science of Learning*, vol. 10, no. 29, 2025.

[14] M. Cukurova, "The interplay of learning, analytics, and artificial intelligence in education: A vision for hybrid intelligence," *British Journal of Educational Technology*, vol. 56, no. 2, pp. 469–488, 2025.

[15] IIT Madras B.S Degree Programme Channel. (August 2023). Deep learning output functions and loss functions. [Online]. Available: https://www.youtube.com/watch?v=1hefEWZHvJg&list=PLZ2ps__7DhBZVxMrSkTIcG6zZBDKUXCnM&index=24

[16] DeepLearningAI Channel. (April 2023). ChatGPT prompt engineering for developers: A short course from OpenAI and DeepLearning.AI. [Online]. Available: https://www.youtube.com/watch?v=H4YK_7MAckk

[17] MIT News. (November 2024). New AI tool generates realistic satellite images of future flooding. *MIT*. [Online]. Available: https://news.mit.edu/2024/new-ai-tool-generates-realistic-satellite-images-future-flooding-1125

[18] A. Bangor, P. T. Kortum, and J. T. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," *Journal of Usability Studies*, vol. 4, no. 3, pp. 114–123, 2009.

Copyright © 2025 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).