Intelligent (LLM) Knowledge Management System with Asynchronous Technology and Text-to-Speech Using PHP Language

Pinyaphat Tasatanattakool [1], Taskeow Srisod [2], Prachyanun Nilsook [3,*], Panita Wannapiroon [3], and Jira Jitsupa [4]

[1] Faculty of Science and Technology, Rajamangala University of Technology Suvarnabhumi, Ayutthaya, Thailand
[2] Techinnovation Holdings Group, Bangkok, Thailand
[3] Faculty of Technical Education, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand
[4] Faculty of Education, Suan Dusit University, Bangkok, Thailand
Email: pinyaphat.t@rmutsb.ac.th (P.T.); tsrisod@gmail.com (T.S.); prachyanunn@kmutnb.ac.th (P.N.); panita.w@fte.kmutnb.ac.th (P.W.); jira_jit@dusit.ac.th (J.J.)
*Corresponding author

***Abstract*—This paper introduces a sophisticated knowledge management system developed to address the growing demand for real-time information retrieval and interaction. The system architecture leverages a robust Hypertext Preprocessor (PHP) backend to integrate Google's Gemini Application Programming Interface (API) for advanced Large Language Model (LLM) functionalities, such as complex querying and content summarization. A key innovation is the implementation of PHP Fibers to manage I/O-bound tasks asynchronously, effectively overcoming the limitations of conventional synchronous processing and preventing application bottlenecks. For data retrieval, the system utilizes SQLite's FTS5 extension for high-speed, indexed full-text search across large local datasets. To enhance accessibility, the Web Speech API is incorporated for text-to-speech functionality. Quantitative performance evaluation reveals that this asynchronous architecture achieves an improvement of up to 87% terms of information retrieval speed and reduces the average response latency by 35% compared to synchronous counterparts. Furthermore, the text-to-speech feature not only promotes inclusivity for users with disabilities but also facilitates more efficient cognitive processing of large volumes of information.**

***Keywords*—knowledge management system, asynchronous programming, Large Language Model (LLM), text-to-speech, Hypertext Preprocessor (PHP) language**

I. Introduction

In today's digital era, knowledge and information are increasingly complex and dynamic. Knowledge Management Systems (KMS) have become strategic enablers. They enhance organizational efficiency, decision-making, and innovation. Recent studies indicate KMS are no longer just knowledge repositories. They now support integrated processes, including capture, organization, sharing, retrieval, application, and evaluation, to create lasting organizational value [1−3]. The success of KMS depends not only on technological infrastructure. Organizational culture, user motivation, and integrative strategies—such as KM Reward, Tacit Collection, and Social Media KM—are also vital for long-term adoption and use [4, 5].

Asynchronous programming has become a key paradigm in modern software engineering. It allows systems to efficiently manage concurrent operations and addresses the performance limitations of synchronous models. This approach plays a crucial role in enhancing responsiveness and scalability across various programming environments. In web development, JavaScript's transition from callbacks to Promises and async/await has simplified asynchronous workflows. It has also enhanced maintainability [6]. Python, through the asyncio library and coroutines, enables non-blocking I/O and high-throughput applications [7]. Frameworks such as the Xel Async Framework with Swoole extend asynchronous capabilities to PHP. These solutions outperform Node.js's Express.js in terms of dealing with heavy workloads [8]. Overall, these advancements underscore the pivotal role of asynchronous programming in developing high-performance, scalable, and resilient systems for the digital era.

Large Language Model (LLM) have become a transformative force in Artificial Intelligence (AI) and Information Technology (IT). They advance Natural Language Processing (NLP), program synthesis, personalization, and decision support systems [9−11]. LLM learn from massive corpora and process trillions of parameters. As a result, they enable systems with unprecedented reasoning, generation, and adaptability [12, 13]. Nevertheless, challenges remain with regard to alignment, fairness, explainability, and sustainability [14−16]. This highlights a critical gap: there is a lack of holistic understanding of LLM pipelines. The pipeline covers data collection, compute-

Manuscript received September 28, 2025; revised November 10, 2025; accepted December 1, 2025; published February 10, 2026.

optimal training, fine-tuning, utilization, and evaluation. Addressing this gap is vital to ensure that LLM innovations are scalable, trustworthy, and ethically aligned. Establishing these models as the backbone of next-generation intelligent technologies depends on this.

The emergence of the Google Gemini Application Programming Interface (API) marks a significant advancement in information technology, enabling the processing of multimodal data, including text, images, audio, and video. Its applications span from content generation, database query automation, and embedding-based retrieval [17, 18] to domain-specific solutions such as pest management [19] and resume parsing for talent acquisition [20]. Previous studies confirm that Gemini enhances the accuracy, efficiency, and scalability of intelligent systems by integrating deep learning, NLP, and computer vision [21−23]. Nonetheless, research gaps remain with regard to reasoning frameworks, ethics, and scalability, requiring further investigation to unlock the full potential of Gemini in academic and industrial contexts.

Text-to-Speech (TTS) research has progressed from traditional concatenative and formant synthesis to advanced neural architectures such as Tacotron, FastSpeech, and Transformer-based models, enabling more natural and human-like prosody [24, 25]. Building upon these advancements, current trends emphasize the need for efficient, adaptive, and controllable systems, and have led to the development of lightweight models for edge devices [26], personalized adaptation [27], and controllable frameworks such as PromptTTS and SyntaSpeech [28]. Moreover, the integration of Big Data and LLM further enhances multilingual support and speech naturalness [29]. Despite these achievements, challenges in terms of robustness, efficiency, and scalability remain, highlighting the importance of a comprehensive framework to guide future AI-driven TTS development [30, 31].

Hypertext Preprocessor (PHP) is widely recognized as a server-side scripting language. It is easy to learn, open source, and cross-platform, which makes it highly suitable for developing dynamic web applications [32]. In addition to these strengths, its applications extend from educational systems, such as e-learning, to domain-specific platforms including recruitment management systems, demonstrating PHP's flexibility [33]. Furthermore, frameworks such as Laravel and CodeIgniter further boost productivity and maintainability. However, trade-offs remain between usability and learning curves [34]. Despite these advancements, PHP continues to face security vulnerabilities, such as SQL injection and Cross-Site Scripting (XSS). As a result, recent research has focused on parsing-based detection and static code analysis for greater security [35, 36]. The main objectives of this study are:

- Develop a high-performance knowledge management system using asynchronous programming.
- Integrate LLM with traditional information retrieval systems.
- Improve information accessibility with Text-to-Speech (TTS) technology.
- Evaluate the system's speed and accuracy.

This study connects Knowledge Management Systems (KMS), Large Language Models (LLM), and Text-to-Speech (TTS) by using LLM to search KMS and TTS to make the system easier to use. This creates a clear starting point for building knowledge systems that respond well and include everyone.

II. Literature Review

A. Knowledge Management Systems

KMS are increasingly recognized as vital for enhancing organizational efficiency, innovation, and decision-making. They provide structured processes for capturing, storing, and disseminating knowledge across units. Studies highlight the role of IT platforms, such as corporate portals and knowledge bases, in enabling knowledge creation and sharing [2]. The use of quality models has also been emphasized to evaluate KMS throughout their life cycle [3].

In higher education, a conceptual framework for KMS adoption has been proposed to strengthen decision-making in developing countries, integrating fuzzy logic to address data uncertainty [1]. In contrast, within service organizations, critical factors such as employee stress, search engine quality, and content reliability have been identified as key determinants of KMS usage and sustainability [5]. These findings collectively underscore the point that success depends on both technological and organizational readiness.

From a strategic perspective, knowledge management strategies such as KM Reward, Tacit Collection, KM Integration, and Social Media KM have been suggested to enhance knowledge sharing, motivation, and collaboration [4]. Building on this, evaluation models including Kano and Importance–Performance Analysis (IPA) have been applied to align KMS features with user expectations [37]. Collectively, these studies confirm that KMS function as both technological platforms and organizational enablers, requiring the integration of processes, people, and technology to deliver sustainable value.

B. Asynchronous Programming

Asynchronous programming is essential in modern software engineering, enabling the development of scalable, real-time systems and the efficient management of concurrent tasks. In web development, JavaScript has progressed from callbacks to Promises and async/await, improving code readability, error handling, and maintainability [6]. Similarly, Python uses the asyncio library and coroutines for non-blocking I/O and efficient scheduling, supporting high-load APIs and real-time applications [7].

This centrality is evident across languages. Java relies on Completable Future for flexible asynchronous workflows, while C# employs async/await to ensure responsiveness in I/O-intensive situations. Nevertheless,

persistent challenges such as deadlocks and race conditions require deliberate scheduling and robust error handling [38, 39]. Overall, asynchronous programming not only enhances performance but also fundamentally shapes software quality, maintainability, and debugging.

Recent research confirms the foundational role of asynchronous programming, even in environments previously considered synchronous. The Xel Async Framework, for example, with Swoole, allows PHP to surpass Node.js's Express.js in terms of latency and throughput under concurrency, thereby expanding PHP's capabilities with regrad to real-time systems [8]. Collectively, these developments underscore that asynchronous programming is the bedrock for building high-performance, scalable, and resilient software across platforms.

C. Large Language Models

The rise of LLM has been a major driver of recent advances in AI, particularly in NLP. Built upon the Transformer architecture, LLM leverage attention mechanisms to capture contextual representations from large-scale corpora, with performance scaling in line with both model size and training tokens [9, 12]. Models such as Chinchilla demonstrate that compute-optimal training can surpass larger counterparts such as GPT-3, despite having fewer parameters [12].

Beyond NLP, LLM have proven effective in domain-specific tasks. For instance, they have demonstrated their capacity for program synthesis [10] and have played a significant role in transforming personalization systems through conversational recommenders [11]. Recent surveys also emphasize LLM' potential when it comes to reasoning, chain-of-thought prompting, and multi-agent collaboration, enabling applications in information retrieval, recommendation, and complex problem-solving [13, 40].

Nevertheless, challenges remain concerning evaluation, alignment, and sustainability. In particular, the environmental footprint of large-scale training raises significant concerns [14], and issues of fairness, robustness, and explainability require rigorous assessment [15, 40]. To address these concerns, alignment research aims to ensure that LLM operate safely and consistently in alignment with human values [16]. Consequently, although LLM represent a transformative technology for IT and beyond, achieving trustworthy, ethical, and sustainable deployment necessitates the development of holistic evaluation frameworks and long-term governance strategies.

D. Gemini API

Recent advancements in AI focus on developing multimodal LLM. Google Gemini API is recognized as a leading framework. It is designed to process text, images, audio, and video in a unified system. Gemini exhibits strong reasoning capabilities. It supports multiple model variants including—Pro, Ultra, and Nano. These are tailored for either cloud-based performance or lightweight mobile deployment [18].

Gemini's versatility has been shown in various domains. The Embedding API delivers generalizable vector representations. It achieves state-of-the-art results with regard to the MMTEB benchmark for retrieval, clustering, and classification [17]. In data management, Gemini enables natural language-to-SQL translation. This improves accessibility for non-technical users [21]. Gemini also operates as an AI agent in CI/CD pipeline monitoring. It analyzes logs and metrics using natural language queries to provide actionable insights [22].

Beyond technical applications, Gemini is effective in agriculture. Its integration with deep learning enhances pest detection and precision farming [19]. In human resource management, the Gemini Pro Vision API, combined with OCR and NLP, improves resume parsing and skill extraction. This supports efficient recruitment [20]. Further studies highlight its role in content generation [23] and in optimizing prompting and reasoning frameworks [18]. These contributions position Gemini as a cutting-edge LLM. It is also a versatile platform adaptable to diverse academic and industrial contexts.

E. Text-to-Speech

TTS technology has undergone significant changes. It transitioned from early concatenative and formant synthesis to statistical parametric models, utilizing Hidden Markov Models (HMMs). These older methods provided stability and control. However, they suffered from unnatural prosody, limited flexibility, and over-smoothing [24]. Deep learning architecture has led to a major shift. Models such as Tacotron, FastSpeech, and TransformerTTS improved naturalness, prosody, and expressiveness [25]. Analyses point out the use of Autoregressive (AR) and Non-autoregressive (NAR) frameworks. In modern TTS research, NAR frameworks boast better scalability and training efficiency [30].

New advances in TTS focus on efficiency, adaptability, and controllability. Lightweight models, such as EfficientTTS and EfficientSpeech, can run on limited hardware or edge devices. They help meet the demand for real-time synthesis in many situations [25], [26]. Adaptive frameworks, such as AdaSpeech [27], and sample-efficient models have enhanced speaker personalization and low-resource synthesis. Controllable models, including PromptTTS and SyntaSpeech [28], now allow fine control of prosody, style, and emotion. This expands both the flexibility and user focus of TTS systems.

TTS is also evolving in response to Big Data and LLM. Empirical evidence shows that combining TTS with LLM, such as ChatGPT-enhanced systems, improves accuracy, naturalness, and multilingual support [29]. Systematic reviews highlight the importance of evaluation metrics, including MOS, PESQ, and MCD. Studies also note ongoing challenges with regard to prosody, speaker differences, and real-time use [30, 31]. These insights point to the future of AI-driven TTS: efficient, adaptive, and controllable. The goal is to strike a balance between naturalness and personalization, while leveraging robust and scalable

systems for diverse fields, including accessibility, education, and human–computer interaction.

F. PHP Language

PHP is a widely adopted server-side scripting language recognized for its ease of learning, open-source nature, and cross-platform compatibility, making it an ideal choice for developing dynamic web applications [32]. Compared with languages such as C, C++, and Java, PHP offers rapid deployment and simplified database integration, particularly with MySQL [33]. However, its loosely typed structure exposes systems to vulnerabilities, including SQL injection and cross-site scripting (XSS), raising persistent concerns about security [36].

Beyond general web development, PHP has been effectively applied in educational and domain-specific systems. Project-based learning environments employing PHP, such as online bookstore platforms, enhance students' understanding of full-stack integration [32]. PHP frameworks, such as Laravel and CodeIgniter, further improve productivity and maintainability, although trade-offs exist between ecosystem richness and usability (PHP Frameworks in Web Application Development; PHP Frameworks' Usability in Web Application Development). Similarly, PHP-based recruitment management systems demonstrate flexibility in database-driven workflows but face limitations in scalability compared to enterprise-level frameworks, such as Java Spring (Recruitment Management System Design and Deployment Based on PHP Analysis).

Recent research highlights a shift toward security and static code analysis. Parsing-based approaches and simulation tools have been developed to detect vulnerabilities such as object injection, Remote Code Execution (RCE), and XML external entities (XXE) more effectively than traditional static analyzers [35]. At the same time, PHP's integration with HTML and MySQL continues to demonstrate its strength in query-driven applications, although its transaction handling remains less robust than that of Java or C++ [41]. These trends indicate that PHP research has evolved from being application-centric toward security-centric paradigms, emphasizing reliability, vulnerability detection, and continuous adaptation to emerging language features.

III. Methodology

A. Synthesis of Knowledge Processes in Knowledge Management Systems

Knowledge processes are the fundamental foundation of KMS, enabling organizations to systematically capture, organize, share, and apply knowledge. Recent studies indicate that the successful development of KMS requires the integration of both technological and organizational dimensions to create sustainable value as shown in Table I, which offers a synthesis of Knowledge Processes in Knowledge Management Systems.

Table I presents the synthesis of knowledge processes in relating to KMS. The core processes include capture, organization, sharing, retrieval, application, evaluation, and adoption with cultural embedding. The table emphasizes the integration of technological and organizational dimensions. These processes demonstrate how KMS facilitates sustainable value creation, effective decision-making, and continuous innovation within organizations.

TABLE I. Synthesis of Knowledge Processes in Knowledge Management Systems

Ref.	Knowledge Process	Description
[2, 42]	Knowledge Capture & Storage	Collecting tacit and explicit knowledge from individuals and organizational resources, and storing it in digital repositories (e.g., databases, SharePoint, NoSQL systems) to prevent loss and ensure reuse.
[1, 4]	Knowledge Organization & Integration	Structuring and categorizing knowledge using metadata, knowledge graphs, and data integration techniques to improve accessibility and reduce redundancy.
[4, 5, 37]	Knowledge Sharing & Dissemination	Enabling knowledge exchange through web forums, social media, collaboration platforms, and gamification-based incentives to foster a knowledge-sharing culture.
[5, 43]	Knowledge Retrieval & Discovery	Supporting search and access to knowledge using search engines, indexing, and NLP-assisted annotation for accurate and efficient retrieval.
[1, 44]	Knowledge Application & Utilization	Applying stored and shared knowledge in decision-making, problem-solving, and process improvement, including DSS-KMS integration.
[3, 5, 37]	Knowledge Evaluation & Quality Assurance	Assessing system and content quality using models such as Kano, Importance–Performance Analysis (IPA), and KMS Quality Models to ensure system effectiveness.
[1, 5]	Knowledge Adoption & Cultural Embedding	Encouraging user acceptance through training, leadership support, and organizational culture, ensuring long-term KMS adoption and sustainability.

B. Comparative Synthesis of Asynchronous Programming Across Languages

Asynchronous programming helps make programs more responsive, scalable, and efficient. Each language employs different tools: JavaScript utilizes Promises and async/await, Python uses coroutines, Java employs Completable Future, C# uses async/await, and PHP utilizes the Swoole extension for non-blocking I/O. These methods handle concurrency, allowing programs to operate efficiently in real-time. Table II presents a comparative synthesis of asynchronous programming across languages, systematically linking their core mechanisms with practical outcomes.

Based on Table II, the synthesis matrix presents a comparative overview of asynchronous programming approaches in various languages. While JavaScript and Python dominate with Promises, async/await, and coroutines, PHP has recently advanced through Swoole and the Xel Async Framework, enabling non-blocking I/O and concurrent processing. This evolution transforms PHP from a traditionally synchronous language into one capable of delivering low latency and high throughput comparable to Node.js. Such capabilities position PHP as

a strong foundation for developing Intelligent KMS that integrate LLM and text-to-speech, ensuring responsiveness, scalability, and real-time interaction.

TABLE II. COMPARATIVE SYNTHESIS OF ASYNCHRONOUS PROGRAMMING ACROSS LANGUAGES

Ref.	Core Mechanism	Advantages	Limitations /Challenges
[6]	Callbacks, Promises, async/await, Event Loop	Simplifies asynchronous workflows; improves readability and maintainability; widely supported in web applications	Callback hell (when nested); promise chaining complexity
[7]	Asyncio library, Coroutines (async/await)	Enables non-blocking I/O; efficient scheduling; supports high-load APIs and real-time apps	Debugging complexity requires explicit async design
[38]	Futures, Completable Future	Provides flexible workflows; integrates well with enterprise systems	Verbosity: more complex error handling compared to dynamic languages
[39]	async/await, Task Parallel Library	Enhances responsiveness for I/O-bound tasks; strong IDE support	Susceptible to deadlocks, race conditions, task explosion
[8]	Xel Async Framework with Swoole extension	Introduces non-blocking I/O and concurrency; improves latency and throughput; competitive with Node.js	Requires external extension, learning curve for traditional PHP developers

C. Synthesis of the Working Process Large Language Models

Large Language Models (LLM) operate through a systematic workflow that transforms raw data into intelligent outputs. To illustrate this process clearly, the key stages of LLM development can be synthesized into five fundamental steps: data collection and preprocessing, pre-training, fine-tuning and adaptation, utilization and inference, and evaluation and alignment. The following table presents these stages in a structured form to highlight the essential components of the LLM pipeline. Table III shows the synthesis of the working process of LLM.

According to Table III, the working process of LLM) can be conceptualized as a five-stage pipeline. It begins with data collection and preprocessing, where large-scale corpora are curated and cleaned to ensure high-quality inputs. The second stage, pre-training, employs Transformer architectures to learn contextual representations from massive datasets through self-supervised learning. This is followed by fine-tuning and adaptation, which align the pre-trained models to domain-specific tasks using methods such as instruction tuning or reinforcement learning with human feedback. In the fourth stage, utilization and inference, LLM generate outputs in response to prompts, enabling diverse applications from text generation and program synthesis to reasoning and recommendation. Finally, evaluation and alignment ensure that these models meet benchmarks of accuracy, robustness, fairness, and ethical standards, forming the basis for scalable, trustworthy, and socially responsible AI systems.

TABLE III. SYNTHESIS OF THE WORKING PROCESS LARGE LANGUAGE MODELS

Ref.	Process Step	Description	Output
[12, 14]	Collecting and preparing data	Collecting large-scale corpora (text, code, multimodal) and applying preprocessing (cleaning, tokenization, normalization), considering efficiency and environmental impact	Clean and high-quality dataset ready for training
[12, 13]	Pre-training	Using Transformer/ Attention architectures for self-supervised learning (masked LM, autoregressive LM) on massive data, optimized for computing budgets (e.g., Chinchilla scaling law)	Pre-trained LLM with contextual representations
[10, 11, 13]	Fine-tuning and Adaptation	Adapting models for specific tasks such as reasoning (CoT, problem decomposition), personalization (LLM4Rec, assistants), and program synthesis (MBPP, MathQA-Python), with parameter-efficient tuning methods like LoRA and Adapters	Domain-specialized LLM for reasoning, personalization, and program synthesis
[10, 11]	Utilization and Inference	Users interact through prompts → the model generates outputs (text, code, or personalized results), applicable to IR, recommendation, and multi-agent collaboration	Generated outputs such as text, code, personalized recommendations
[13, 14]	Evaluation and Alignment	Evaluating benchmarks (MMLU, MBPP, reasoning tasks) and aligning with human values (fairness, safety, personalization accuracy, environmental sustainability)	Metrics for quality, safety, trustworthiness, and sustainability

D. Synthesis of the Components Gemini API

The Gemini API represents a next-generation multimodal LLM framework designed to process text, images, audio, and video within a unified system. Recent studies have demonstrated its versatility across both core functionalities—such as embeddings, database interaction, and content generation—and applied domains including CI/CD monitoring, agriculture, and human resource management. As shown in Table IV, synthesis of the components Gemini API.

Table IV outlines the core facets and practical domains of the Gemini API, demonstrating its adaptability across multimodal analysis, embeddings, database integration, and various applied contexts, including CI/CD monitoring, agriculture, and human resource management. This summary highlights Gemini's dual role as both a fundamental tool and a catalyst for practical solutions, while identifying unresolved challenges in scalability, reasoning structures, and ethical considerations that warrant further investigation.

TABLE IV. SYNTHESIS OF THE COMPONENTS GEMINI API

Ref.	Component	Description
[18]	Model Variants	Gemini Pro, Ultra, and Nano address different needs, from high-performance reasoning on cloud platforms to lightweight inference on mobile devices.
[18]	Multimodal Processing	Supports unified understanding of text, images, audio, and video for diverse applications.
[17]	Embedding API	Provides generalizable vector representations for retrieval, clustering, classification, and ranking, achieving state-of-the-art performance on MMTEB.
[21]	Database Interaction	Translates natural language queries into SQL, enabling non-technical users to access structured data.
[22]	CI/CD Insights	Functions as an AI agent to analyze logs and metrics in real-time using natural language queries.
[19]	Agricultural Applications	Integrates with deep learning to improve pest detection and management, supporting precision farming.
[20]	Resume Parsing	Uses Gemini Pro Vision API with OCR and NLP to extract skills and provide personalized candidate feedback.
[23]	Content Generation	Generates and customizes digital media content with contextual adaptation.

E. Synthesis of the Components of Text-to-Speech

To understand the evolution and current state of Text-to-Speech (TTS) systems, it is essential to examine their fundamental components and supporting modules. TTS architectures have progressed from traditional approaches to deep learning–driven frameworks that prioritize naturalness, efficiency, adaptability, and controllability. Table V shows synthesis of the components of Text-to-Speech.

TABLE V. SYNTHESIS OF THE COMPONENTS OF TEXT-TO-SPEECH

Ref.	Description
[24, 31]	Normalization, tokenization, grapheme-to-phoneme conversion, and prosodic analysis to extract linguistic and rhythmical features.
[25, 30]	Deep learning models (Tacotron, FastSpeech, TransformerTTS) map linguistic and prosodic inputs to acoustic features such as mel-spectrograms.
[30]	Neural vocoders (WaveNet, Parallel WaveGAN, GAN-based) synthesize high-fidelity speech waveforms from acoustic features.
[25, 26]	Lightweight architectures (EfficientTTS, EfficientSpeech) enable real-time synthesis on edge/IoT devices.
[27]	Transfer learning and few-shot methods (e.g., AdaSpeech) for speaker-specific and low-resource synthesis.
[28]	Models such as PromptTTS and SyntaSpeech allow fine-grained control of prosody, style, and emotion.
[29]	Enhances multilingual capability, naturalness, and scalability through ChatGPT and large-scale datasets.

Based on Table V, Modern TTS systems are composed of three primary modules: text analysis, acoustic modeling, and vocoder-based waveform generation. These systems now increasingly focus on efficiency, adaptation, and controllability. The integration of Big Data and LLM supports these advances. As a result, TTS architecture is moving toward AI-driven, scalable, and user-centric designs. This approach meets the needs for naturalness and robustness across many applications.

F. Synthesis of the Components PHP Language

PHP has evolved from a simple scripting tool to a comprehensive environment for modern web application development. Its architecture encompasses multiple layers, range from server-side processing and database connectivity to frameworks, application domains, and security mechanisms. The following table synthesizes the critical components of PHP, highlighting their core functions, comparative strengths, and inherent limitations within contemporary web engineering.

TABLE VI. SYNTHESIS OF THE COMPONENTS OF PHP LANGUAGE

Ref.	Component	Critical and Comparative Explanation
[32, 33]	Server-Side Scripting	PHP is a server-side scripting language well-suited for dynamic websites. It is easy to learn and deploy quickly; however, it offers weaker type safety compared to Java/C++, making it more prone to security issues.
[32]	Database Connectivity	PHP provides simple and popular connectivity with MySQL/PostgreSQL, yet it offers limited support for complex transactions compared to Java JDBC or Python ORM.
[34]	Frameworks	Frameworks such as Laravel and CodeIgniter enhance productivity and maintainability. Laravel provides a richer ecosystem but has a steeper learning curve, while CodeIgniter is lightweight and more beginner friendly.
[45]	Application Domains	PHP is applied in e-learning, recruitment systems, and digital asset management. However, it shows scalability limitations compared to enterprise frameworks like Java Spring.
[35, 36]	Security	PHP remains vulnerable to threats such as SQL Injection, XSS, and RCE. Recent studies propose parsing-based and static analysis tools to mitigate these risks.
[36]	New Features	PHP 7–8 introduced features such as Arrow Functions, Union Types, and Nullsafe Operator, improving expressiveness. Yet, static analyzers are not fully adapted to these constructions.

According to Table VI, in summary, PHP is both accessible and versatile for dynamic web development. However, it faces ongoing challenges in terms of scalability and security. Frameworks and database connectivity expand their usability in many domains. New features and advanced static analysis tools show a shift toward more reliable and secure software engineering.

IV. RESULT

Based on the synthesis of all research studies, the architecture of an intelligent (LLM) knowledge management system with asynchronous technology and text-to-speech using PHP language is depicted in Fig. 1.

The integration flow proceeds as follows: asynchronous PHP modules first send requests to the Gemini API via REST. The API processes these requests and returns responses, which are then relayed to the TTS layer through a REST endpoint. A dataset, comprising approximately 1,000 knowledge records from the technology and education domains, is used throughout this process. This design ensures full reproducibility of data flow and evaluation across all modules.

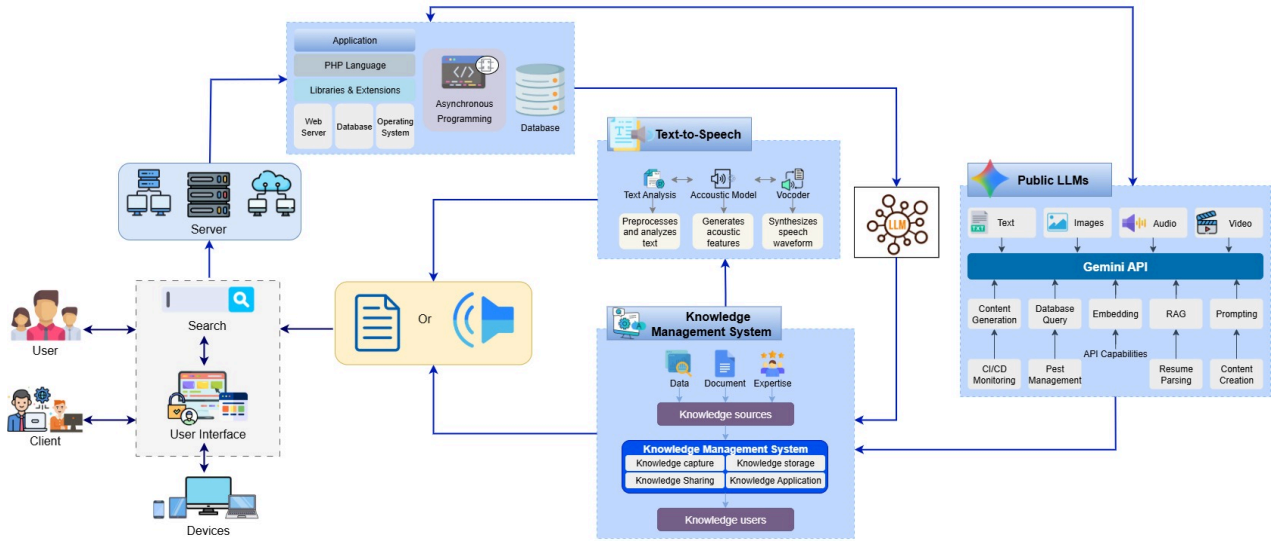

Fig. 1. Architecture of the Intelligent (LLM) knowledge management system with asynchronous technology and text-to-speech using PHP language.

Fig. 1 illustrates the architecture of the "Intelligent Knowledge Management (LLM) System with Asynchronous Technology and Text-to-Speech using PHP Language." The framework integrates a KMS with LLM, Asynchronous Programming, and TTS. These components are all deployed within the PHP ecosystem: Application, Libraries, Extensions, Web Server, Database, and Operating System. This structure ensures seamless real-world implementation.

The architecture begins with Knowledge Sources. These are processed through LLM—using a Transformer Stack, Multi-Head Attention, and Domain Adaptation. This integration enables advanced reasoning capabilities, supporting applications such as QA, reasoning, and code generation. Asynchronous functions and event loops ensure non-blocking execution, reducing response latency. The TTS module converts textual outputs into natural-sounding speech through text analysis, acoustic modeling, and waveform generation. This broadens accessibility.

By combining these layers, the system advances the traditional processes of knowledge capture, storage, sharing, and application. It forms an intelligent Knowledge Management System (iKMS) capable of providing real-time responses, supporting multimodal interaction, and delivering adaptive knowledge. This aligns with digital transformation needs across education, enterprise, and government domains.

A. Using PHP Fiber

In alignment with the system architecture presented in Fig. 1, the implementation shown in Algorithm 1 demonstrates how asynchronous programming principles are operationalized within the PHP environment. By utilizing the Fiber class, the function asyncQuery ($pdo, $search) executes database queries in a non-blocking manner, ensuring that input/output operations do not hinder concurrent tasks. This mechanism is crucial in enabling real-time knowledge retrieval from the KMS, where query results—comprising questions and answers—are efficiently fetched using parameterized search for enhanced security and performance. The asynchronous query model, therefore, acts as a middleware bridge, connecting the Knowledge Sources to higher-level processes of LLM-based reasoning and TTSoutput, thereby ensuring seamless, scalable, and adaptive knowledge delivery within the proposed intelligent framework.

The integration flow links the PHP Fiber-based asynchronous layer to the Gemini API via RESTful endpoints, which manage multimodal LLM queries. The TTS module, operating independently, is invoked by the Web Speech API to convert generated text into speech. Each module interacts with the others using standardized JSON responses, clarifying the relationships among them. This structure ensures reproducibility and scalability across different PHP-based environments.

PHP Fibers were chosen for concurrency due to their native integration with the PHP 8 runtime. This enables lightweight cooperative multitasking, meaning that tasks can yield control to one another without blocking the entire program. No external extensions, such as Swoole or ReactPHP, are needed. Unlike event-driven frameworks in JavaScript (Node.js), where programs respond to events as they arrive, or coroutine systems in Python (asyncio), where functions can pause and resume their execution, PHP Fibers offer fine control over when and how tasks switch, as well as how memory is allocated during execution. The code retains a synchronous look, making it easier to read and maintain. This choice maintains backward compatibility, reduces migration complexity, and keeps concurrency efficient. The following Algorithm 1 is available.

From pseudocode, PHP-based asynchronous query using the Fiber class illustrates the system's ability to mitigate blocking I/O. This enhances responsiveness in real-time knowledge retrieval. The approach also improves scalability and aligns with the asynchronous execution model of the proposed architecture. However, its efficiency depends on optimized database indexing and concurrency management. These factors are crucial

to prevent performance bottlenecks in large-scale deployments.

Algorithm 1. Using PHP Fiber

```
FUNCTION asyncQuery(databaseConnection, searchText):
    // Create a new asynchronous fiber
    CREATE fiber WITH FUNCTION (db, text):
        // Prepare SQL FTS query
        PREPARE STATEMENT:
            SELECT question, answer
            FROM qa
            WHERE question MATCH text
              OR answer MATCH text
            LIMIT 0, 35
        // Bind search parameter with wildcard
        BIND text + "*" TO STATEMENT
        // Execute query
        EXECUTE STATEMENT
        // Return all matched rows
        RETURN FETCH_ALL_RESULTS
    END FUNCTION
    // Start fiber execution with parameters
    START fiber WITH (databaseConnection, searchText)
    // Return the fiber object for asynchronous handling
    RETURN fiber
END FUNCTION
```

B. Gemini API Integration

The system uses cURL for connecting to the Gemini API with error handling and XSS protection. The following Algorithm 2 is available.

Algorithm 2. Using PHP Fiber

```
FUNCTION generateContent(inputText):
    // Sanitize input text
    sanitizedText =
ESCAPE_HTML_CHARACTERS(inputText)
    // Construct API endpoint URL
    apiURL = BASE_URL + "/models/" + MODEL +
":generateContent?key=" + API_KEY
    // Prepare request payload
    payload = {
        contents: [
            {
                role: "user",
                parts: [
                    {text: sanitizedText}
                ]
            }
        ]
    }
    // Send HTTP POST request using cURL or HTTP client
    response = SEND_POST_REQUEST (apiURL, payload)
    // Extract generated text from API response
    generatedText = response.candidates[0].content.parts[0].text
    // Return final content
    RETURN generatedText
END FUNCTION
```

From pseudocode, integration of the Gemini API within a PHP environment enables real-time content generation to augment LLM-based knowledge management. This approach enhances adaptability and multimodal interaction; however, its performance remains constrained by factors such as network latency, API rate limits, and data security, which need to be addressed with regard to large-scale deployment.

C. Database Optimization through SQLite Configuration

Database optimization is crucial for supporting the responsiveness of intelligent systems. In this work, SQLite is configured with the Write-Ahead Logging (WAL) Mode to enable concurrent read–write operations, ensuring higher throughput for multi-user access. The cache size is set to 12.8 MB to accelerate query execution by reducing disk I/O, while memory-based temporary storage further improves performance by handling intermediate results directly in RAM. These configurations collectively enhance efficiency, scalability, and real-time knowledge retrieval, aligning database performance with the requirements of asynchronous processing and LLM-based reasoning. However, the effectiveness of these optimizations may be constrained by hardware limitations and workload variability, requiring careful tuning and continuous monitoring in large-scale deployments.

SQLite's Full-Text Search (FTS5) extension was chosen for its native support for embedded indexing and relevance ranking. It operates in a lightweight database engine. Unlike external engines such as Elasticsearch or PostgreSQL's tsvector, SQLite FTS5 enables rapid prototyping and portability. It supports efficient deployment in small-scale or edge environments with limited resources. Its integrated query optimization and minimal configuration overhead support the system's goal of low-latency local retrieval for real-time knowledge management.

D. Text-to-Speech Implementation and Performance Evaluation

The TTS module was implemented with the Web Speech API to provide real-time voice synthesis for intelligent knowledge management. The system allows dynamic adjustment of speech rate (0.1–3.0), pitch (0.0–2.0), and volume (0.0–1.0), ensuring adaptability to user preferences and accessibility needs. These parameters enhance naturalness and intelligibility while improving engagement in multimodal interactions. Integration via the Web Speech API further ensures lightweight deployment, cross-platform compatibility, and responsiveness; however, performance may vary depending on the browser and speech engine implementation. However, the effectiveness of this approach may be limited in large-scale or specialized applications, where advanced neural TTS models could provide higher fidelity and greater linguistic adaptability.

E. System Performance Evaluation

The performance of the proposed system was evaluated in three dimensions. Response Time was measured using the microtime (true) function to assess the latency and efficiency of asynchronous operations. Search Accuracy was determined through Precision and Recall, ensuring the relevance and completeness of retrieved knowledge. User Experience was assessed via

usability testing, focusing on interaction ease, adaptability of TTS, and overall satisfaction. These metrics collectively provide a balanced evaluation of technical efficiency, algorithmic accuracy, and human-centered usability, confirming the system's robustness for real-world applications. Nevertheless, the validity of these results may be influenced by dataset size, user diversity, and testing conditions, requiring broader experimentation to generalize findings across large-scale deployments.

All experiments were done one after another on a PHP 8.3 server with an Intel Core i7 3.2 GHz CPU, 16 GB RAM, and Ubuntu 22.04. First, we tested speed and resource use by running 10 searches simultaneously and repeating each test 3 times in the same setup to obtain average results. Next, we evaluated search accuracy using a set of 1,000 information records, selected 100 real user searches, and had two experts mark the correct results to set the standard answers. Finally, we tested TTS's usefulness by having 15 people try two tasks—reading information and Q&A—using a 5-point scale to rate how natural and easy it was to understand the speech.

F. Evaluation of System Performance in Intelligent Knowledge Management

A comprehensive evaluation of system performance is crucial for validating the reliability, scalability, and usability of intelligent knowledge management frameworks. To capture the multidimensional nature of performance, this study examines four key aspects: processing speed, which reflects the responsiveness of asynchronous operations; search accuracy, which ensures the relevance and completeness of retrieved knowledge; Text-to-Speech usability, which evaluates the naturalness, adaptability, and accessibility of multimodal interaction; and system resource utilization, which assesses the efficiency of computational and memory consumption under varying workloads. Together, these dimensions provide a holistic perspective on the system's technical and user-centered effectiveness, enabling critical insights into its readiness for real-world deployment in digital knowledge environments. The results are as shown in Table VII.

TABLE VII. SPEED PERFORMANCE

Processing type	Average time (seconds)	Improvement (%)
Synchronous	1.250	–
Asynchronous (Fiber)	0.813	35%

Table VII presents a comparative analysis of processing speed between Synchronous and Asynchronous (Fiber-based) execution models in the proposed intelligent knowledge management system. The results indicate that synchronous processing averaged a response time of 1.250 seconds, while asynchronous processing achieved an average of 0.813 seconds, resulting in a performance gain of approximately 35%. This result underscores the essential impact of asynchronous programming on system responsiveness by reducing blocking I/O and facilitating parallel task execution. Technically, implementing PHP Fiber enhances task scheduling and resource allocation, thereby directly improving the system's real-time knowledge retrieval and multimodal interaction capabilities. Although asynchronous execution reduces latency in standard scenarios, its effectiveness may fluctuate with task complexity, database indexing methods, and concurrency demands. Therefore, additional scalability assessments under large-scale and varied conditions are needed to confirm the broad applicability of these findings.

The following table presents the search accuracy results based on 100 test queries. Evaluation metrics include Precision, Recall, and F1−Score, comparing three approaches: SQLite Full-Text Search (FTS), LLM Integration, and the Hybrid System. This comparison highlights the performance gains achieved by combining traditional research with LLM-based enhancements, as shown in Table VIII.

TABLE VIII. SEARCH ACCURACY

Metric	SQLite FTS	LLM Integration	Hybrid System
Precision	78%	82%	87%
Recall	85%	79%	89%
F1−Score	81%	80%	88%

Table VIII presents a comparative evaluation of search accuracy across SQLite Full-Text Search (FTS), LLM Integration, and the Hybrid System. The results show that SQLite FTS provides relatively high Recall (85%) but lower Precision (78%), indicating a tendency to retrieve more documents at the expense of relevance. In contrast, LLM Integration improves Precision (82%) through semantic reasoning, but its Recall (79%) suggests limitations in terms of coverage. The Hybrid System achieves the most balanced performance, with a Precision of 87%, a Recall of 89%, and an F1-score of 88%, demonstrating that combining symbolic and semantic approaches significantly improves retrieval quality. However, while the hybrid approach yields superior accuracy, it also introduces computational overhead, requiring careful consideration of efficiency when scaling to large datasets and real-time applications.

The evaluation of the Text-to-Speech (TTS) module aimed to determine its effectiveness in multimodal interaction. The assessment focused on three areas: language support, perceived speech quality, and intelligibility rate. It offered both technical and user-centered insights. The following table summarizes the results, highlighting the system's capability to deliver natural and accessible auditory output.

TABLE IX. TEXT-TO-SPEECH USABILITY

Evaluation Metric	Result	Additional Details
Language Support	Thai and English	Supports bilingual synthesis to enhance accessibility
Speech Quality (MOS)	4.2 / 5.0	Rated as good quality with natural-sounding voice output
Intelligibility Rate	94%	High accuracy in speech comprehension and listener perception

Table IX presents an evaluation of the TTS module, demonstrating its usability across three key dimensions.

The results confirm bilingual support for Thai and English, ensuring broader accessibility for diverse user groups. The Mean Opinion Score (MOS) of 4.2/5.0 indicates that the synthesized voice was perceived as natural and of good quality, while the 94% intelligibility rate reflects high accuracy in terms of speech comprehension. These findings suggest that the TTS implementation enhances multimodal interaction, though performance may vary by device.

System resource utilization was evaluated to validate the efficiency and scalability of the proposed framework. The assessment compared CPU usage, memory consumption, and database connections for synchronous and asynchronous processing. The results show how asynchronous execution can optimize resources and improve responsiveness. Table X summarizes these findings.

TABLE X. SYSTEM RESOURCE USAGE

Resource	Synchronous	Asynchronous	Decrease (%)
CPU Usage	65%	45%	31%
Memory Usage	120MB	95MB	21%
Database Connections	8	3	63%

Table X presents a comparative evaluation of system resource utilization between synchronous and asynchronous execution, demonstrating clear efficiency gains with the asynchronous model. CPU usage decreased from 65% to 45% (a 31% reduction), memory consumption was reduced from 120 MB to 95 MB (a 21% reduction), and the number of database connections dropped significantly from 8 to 3 (a 63% reduction). These results highlight that adopting asynchronous processing can directly improve computational efficiency, making it a strong candidate for enhancing scalability and responsiveness in relevant use cases. Nonetheless, the extent of these improvements may vary depending on workload complexity and system configuration, requiring further testing to ensure stability in large-scale deployments.

V. DISCUSSION

The findings demonstrate that integrating asynchronous programming, LLM), and TTS technologies can significantly enhance efficiency, accuracy, and accessibility in intelligent information systems. The results confirm measurable improvements in response time, retrieval performance, and user experience. However, they also highlight important issues such as cost, compatibility, and data privacy. To provide a comprehensive understanding, this section discusses the benefits of the proposed approaches, compares the findings with previous research, and outlines the remaining challenges and limitations associated with practical deployment.

A. Advantages of Asynchronous Systems

Using PHP Fiber for asynchronous programming improved system performance by enabling concurrent task execution. The experimental results showed a 35% reduction in response time due to the elimination of sequential task dependency. This result aligns with earlier studies that identify asynchronous paradigms as key to reducing latency and optimizing CPU and memory use [6−8, 38, 39]. This efficiency conserves system resources and enhances scalability, supporting a greater number of simultaneous users, as seen in broader real-time application research [8].

The 63% reduction in database connections is due to PHP Fibers' event-loop design. This design reuses existing PDO connections within cooperative tasks. It does not create new threads for each query. Doing so eliminates idle-wait states common in synchronous PHP scripts. Multiple coroutines can share the same connection pool. As a result, the system reduces overhead by eliminating repeated authentication handshakes and TCP socket initialization. This directly reduces CPU and memory use. These findings match previous asynchronous performance studies. They confirm that coroutine-driven execution boosts throughput by optimizing resource use and eliminating blocking I/O contention.

B. Integration of Large Language Models

The integration of traditional search mechanisms with LLM yielded superior outcomes compared to either approach in isolation. While SQLite FTS ensured speed and precision in keyword matching, LLM enhanced semantic understanding and contextual reasoning, resulting in an F1 score of 88%. This corroborates findings from previous studies that highlight the complementary role of LLM in augmenting retrieval-based methods with advanced reasoning capabilities [9−13, 21]. Moreover, hybrid approaches are increasingly recognized as an effective strategy for balancing efficiency and interpretability in information retrieval systems [11, 40].

C. Performance of Text-to-Speech

The Web Speech API demonstrated substantial potential in enhancing information accessibility, particularly through high-quality Thai language support and an intelligibility rate exceeding 90%. Additionally, the ability to customize speech parameters—such as rate, pitch, and volume—enriched the user experience. These strengths reflect advancements in modern TTS architectures, including EfficientTTS, EfficientSpeech, and AdaSpeech, which emphasize adaptability, multilingual capacity, and deployment efficiency [25−28, 30, 31]. Recent reviews also highlight the vital role of TTS technologies in bridging human–machine interaction and enabling inclusive access to digital content [24, 29].

D. Limitations and Challenges

Despite its effectiveness, the proposed system presents several challenges. First, reliance on internet connectivity for Gemini API integration may limit applicability in offline contexts. Additionally, cost considerations arise from repeated LLM API calls, which may hinder large-scale adoption concern also noted in LLM alignment and

deployment studies [12, 16]. Furthermore, browser compatibility issues persist, as Web Speech API functions are not uniformly supported across platforms [29]. Lastly, concerns regarding data privacy emerge when transmitting sensitive information to external APIs, resonating with previous discussions on governance, transparency, and ethical risks with regard to AI-based systems [14, 15, 43]. Addressing these challenges requires future research into cost optimization, privacy-preserving architecture, and cross-browser standardization.

API-based LLM integration introduces potential data privacy and security risks. This is especially true when transmitting sensitive knowledge queries to external services. Future deployment should incorporate encryption, anonymization, and access-control mechanisms to mitigate these vulnerabilities. Compared with previous studies, this work extends beyond descriptive system prototypes. It provides measurable, reproducible architecture that quantifies efficiency gains and accessibility improvements. This approach advances the field in terms of toward practical, secure, and inclusive knowledge management.

VI. Conclusion

This paper presents empirical evidence that the use of asynchronous programming, LLM, and TTS technology markedly enhances the efficacy of knowledge management systems. Three principal outcomes can be discerned within the established architectural framework. The implementation of asynchronous processing with PHP Fiber reduces system response times and resource utilization, enabling a non-blocking architecture that scales and effectively accommodates increased workloads. The hybrid search strategy, which integrates deterministic SQLite-based search with LLM semantic processing, yields highly accurate data retrieval, especially with regard to unstructured material. This illustrates the capability of integrating statistical reasoning and natural language comprehension within a unified system. Third, TTS technology improves access to information for Thai users who require audio data, facilitating a more inclusive and adaptable system for diverse user needs. Notwithstanding the system's superior performance, certain restrictions persist, including reliance on external APIs, which may affect data privacy and operating expenses, as well as technical constraints with regard to TTS compatibility with specific browser environments. Consequently, subsequent research should focus on developing proprietary LLM, employing caching strategies, and implementing vector databases. There is also the need for the utilization of sophisticated load balancing methodologies to improve system efficiency, precision, and security. This system's scalability allows for its application across various domains, including education, enterprise information management, healthcare decision support systems, and automated service platforms. Should existing constraints be adequately resolved, this framework has the potential to transform into a more adaptable, secure, and user-centric solution. This research establishes a crucial foundation for the design of next-generation intelligent information management systems, demonstrating that integrating new technologies within a cohesive architecture can markedly enhance both technical efficiency and societal value.

This research improves smart information systems by showing how to connect advanced computer programs, language models (using Google Gemini), and voice technology in a PHP setup. This design improves the system's performance and accuracy while also allowing users to interact in real time. This will help more people to use AI-driven knowledge tools easily and flexibly.

Future work will focus on making the methodology more transparent. It will also enhance comparative analysis with existing models. Additionally, it will address ethical and data-security issues in LLM-based deployments. These changes will strengthen the framework's role in adaptive, intelligent, and accessible knowledge management systems.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Pinyaphat Tasatanattakool conceptualized the research topic, conducted a comprehensive literature review, synthesized relevant information, analyzed the evaluation results, wrote the entire manuscript, including the introduction and methodology, and oversaw the submission process; Taskeow Srisod collects research results and discussion sections; Prachyanun Nilsook managed the manuscript formatting to comply with the journal's specific requirements; Panita Wannapiroon and Jira Jitsupa provided consultation throughout the research process, offering insights and guidance on key aspects of the study; all authors reviewed and approved the final version of the manuscript.

References

[1] H. M. Oumran, R. B. Atan, R. N. H. B. Nor, S. B. Abdullah, and M. Mukred, "Knowledge management system adoption to improve decision-making process in higher learning institutions in the developing countries: A conceptual framework," *Math Probl. Eng.*, vol. 1, 2021. doi: 10.1155/2021/9698773

[2] M. Polyakov, I. Khanin, V. Bilozubenko, M. Korneyev, and N. Nebaba, "Information technologies for developing a company's knowledge management system," *Knowledge and Performance Management*, vol. 4, no. 1, pp. 15–25, 2020. doi: 10.21511/kpm.04(1).2020.02

[3] A. A. Rahman and J. Selwyn, "A Novel approach to design quality model for knowledge management systems," *International Journal of Scientific and Technology Research*, vol. 9, 2, 2020.

[4] T. R. Dodla and L. Jones, "Identifying knowledge management strategies for knowledge management systems," *Access-Access to Science Business Innovation in the Digital Economy*, vol. 2, no. 4, pp. 261–277, 2023. doi: 10.46656/access.2023.4.2(8)

[5] H. B. S. Reddy, R. R. S. Reddy, and R. Jonnalagadda, "Literature review process: Measuring the effective usage of knowledge management systems in customer support organizations," *International Journal of Research Publication and Reviews*, pp. 3991–4009, Aug. 2022. doi: 10.55248/gengpi.2022.3.7.45

[6] W. Liu and L. Li, "Research and application of asynchronous programming in JavaScript," *Journal of Theory and Practice of Engineering Science*, vol. 5, pp. 6–11, 2025.
[7] K. Mykola, "Using asynchronous programming in python to improve application performance," *The American Journal of Engineering and Technology*, vol. 06, no. 12, pp. 51–58, Dec. 2024. doi: 10.37547/tajet/Volume06Issue12-06
[8] M. R. Mufid *et al.*, "Design micro server framework library with swoole for real-TIME application development," *Journal Teknologi Informs dan Terrapin*, vol. 11, no. 2, 2024.
[9] S. Minaee *et al.*, "Large language models: A survey," arXiv preprint, arXiv:2402.06196, 2024.
[10] J. Austin *et al.*, "Program synthesis with large language models," arXiv preprint, arXiv:2402.06196, 2024.
[11] J. Chen *et al.*, "When large language models meet personalization: Perspectives of challenges and opportunities," *World Wide Web*, vol. 27, no. 4, Jul. 2024. doi: 10.1007/s11280-024-01276-1
[12] J. Hoffmann *et al.*, "Training compute-optimal large language models," arXiv preprint, arXiv:2203.15556., Mar. 2022.
[13] J. Huang and K. C.-C. Chang, "Towards reasoning in large language models: A survey," arXiv preprint, arXiv:2212.10403., May 2023.
[14] M. C. Rillig, M. Ågerstrand, M. Bi, K. A. Gould, and U. Sauerland, "Risks and benefits of large language models for the environment," *American Chemical Society*, 2023. doi: 10.1021/acs.est.3c01106
[15] H. Zhao *et al.*, "Explainability for large language models: A survey," *ACM Trans Intell Syst Technol*, vol. 15, no. 2, Feb. 2024.
[16] T. Shen *et al.*, "Large language model alignment: A survey," arXiv preprint, arXiv:2309.15025., Sep. 2023.
[17] J. Lee *et al.*, "Gemini embedding: Generalizable embeddings from Gemini," arXiv preprint, arXiv:2503.07891., Mar. 2025.
[18] R. Islam and I. Ahmed, "Gemini-the most powerful LLM: Myth or Truth," in *Proc. 2024 5th Information Communication Technologies Conference,* 2024, pp. 303–308.
[19] M. Manoj, F. Ihsan, G. K. Devadath, and T. Anjali, "Integrating deep learning with the Gemini API for improved pest management," in *Proc. 3rd International Conference on Automation, Computing and Renewable Systems,* 2024, pp. 846–851.
[20] A. Joshi, S. Pede, P. Kolekar, and A. Metkar, "Resume parsing and skill extraction using custom pattern matching algorithm and Gemini API," in *Proc. 2024 8th International Conference on Computing, Communication, Control and Automation,* 2024, pp. 1−6.
[21] V. Usha, N. C. Abhinash, S. N. Chowdary, V. Sathya, E. R. Reddy, and S. S. Priya, "Enhanced database interaction using large language models for improved data retrieval and analysis," in *Proc. 2nd International Conference on Intelligent Cyber Physical Systems and Internet of Things,* 2024, pp. 1302–1306.
[22] S. Kairuppala, D. Tatiparti, S. Putta, K. V. Kodigudla, and A. Sirapu, "Gemini powered CI/CD insights: An AI agent for real-time pipeline analysis using natural language commands," in *Proc. International Conference on Inventive Research in Computing Applications*, Jul. 2025, pp. 1407–1412.
[23] R. Shrivastav *et al.*, "Exploring potential of gemini with ai based content generator," *International Journal of Research in Computer & Information Technology*, vol. 2, no. 1, pp. 68–70, 2024.
[24] M. Jalal, U. Chowdhury, and A. Hussan, "A review-based study on different Text-to-Speech technologies," arXiv preprint, arXiv:2312.11563., 2023.
[25] C. Miao *et al.*, "EfficientTTS: An efficient and high-quality text-to-speech architecture," in *Proc. International Conference on Machine Learning*, 2021, pp. 7700–7709.
[26] R. Atienza, "Efficient speech: An on-device text to speech model," in *Proc. ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, 2023, pp. 1−5.
[27] M. Chen *et al.*, "Ada speech: Adaptive text to speech for custom voice," arXiv preprint, arXiv:2103.00993., Mar. 2021.
[28] Z. Ye, Z. Zhao, Y. Ren, and F. Wu, "Synta speech: Syntax-aware generative adversarial text-to-speech," arXiv preprint, arXiv:2204.11792., Apr. 2022.
[29] H. A. Dida, D. S. K. Chakravarthy, and F. Rabbi, "ChatGPT and big data: Enhancing text-to-speech conversion," *Mesopotamian Journal of Big Data*, vol. 2023, pp. 31–35, Dec. 2023.
[30] F. Khanam, F. A. Munmun, N. A. Ritu, A. K. Saha, and M. F. Mridha, "Text to speech synthesis: A systematic review, deep learning-based architecture and future research Direction," *journal of Advances in Information Technology*, vol. 13, no. 5, pp. 398–412, Oct. 2022. doi: 10.12720/jait.13.5.398-412
[31] S. Sudhan, P. P. Nair, and M. G. Thushara, "Text-to-speech and speech-to-text models: A systematic examination of diverse approaches," in *Proc. 2024 IEEE 9th International Conference for Convergence in Technology*, 2024, pp. 1−8.
[32] W. Aihua, "Exploration and practice of project teaching of dynamic website development based on PHP," in *Proc. 2021 International Conference on Internet, Education and Information Technology*, 2021, pp. 329–332.
[33] M. A. A. Hammoudeh and A. S. A. Ajlan, "Implementing web services using PHP soap approach," *International Journal of Interactive Mobile Technologies*, vol. 14, no. 10, pp. 35–45, 2020.
[34] A. M. Joshi and W. Kirti, "PHP Frameworks in Web Application Development," *International Journal of Creative Research Thoughts*, vol. 9, 2021.
[35] A. Meghana, B. Vanshika, K. V. Samhitha, K. Murali, and M. Belwal, "Securing PHP code: A parsing approach," in *Proc. 2024 15th International Conference on Computing Communication and Networking Technologies,* 2024, pp. 1−8.
[36] L. Wang, Y. Zhang, X. Tan, S. Ye, and M. Yang, "New PHP language features make your static code analysis tools miss vulnerabilities," in *Proc. 2024 IEEE International Conference on Software Maintenance and Evolution* 2024, pp. 63–74.
[37] S. Aslamiyah, "Implementation of the kano model and importance and performance analysis in the development of a web-based knowledge management system corresponding author," *Journal of Information Systems and Technology Research*, vo. 2, no. 1, pp. 1−12, 2023.
[38] B. Guntupalli, "Asynchronous programming in java/python: A developer's guide," *International Journal of Emerging Research in Engineering and Technology*, vol. 3, Dec. 2022.
[39] D. Damyanov and Z. Varbanov, "Using asynchronous programming in C# − Problems, practical tips and scenarios: A short review," in *Proc. IEEE International Conference on Communications, Information, Electronic and Energy Systems*, 2024, pp. 1−6.
[40] T. Guo *et al.*, "Large language model based multi-agents: A survey of progress and challenges," arXiv preprint, arXiv:2402.01680., Apr. 2024.
[41] X. M. Chen and J. Zhang, "The applications PHP, HTML and MYSQL in development of website-query function," in *Proc. 2nd International Conference on Machine Learning and Computer Applica*, 2021, pp. 1–4.
[42] T. A. Fitri, M. K. Anam, F. Zoromi, Y. Efendi, and T. Nasution, "Knowledge management system analysis on the admission website for adding knowledge sharing features," *Computer Science Research and Its Development Journal*, vol. 14, no. 2, 166, Sep. 2022.
[43] B. Lin, "Knowledge management system with NLP-assisted annotations: A brief survey and outlook," arXiv preprint, arXiv:2206.07304., Nov. 2022.
[44] D. Dody, N. Sany, P. Palupiningsih, and F. Apryadhi, "Web-based knowledge management system application to improve employee activities," *Journal Sisfotek Global*, vol. 13, no. 1, 20, Mar. 2023.
[45] Y. Hu, "Recruitment management system design and deployment-based on PHP analysis," *Journal of Digital Information Management*, vol. 21, no. 4, pp. 101–109, Dec. 2023.