International Journal of Scientific Research & Engineering Trends
Volume 11, Issue 3, May-June-2025, ISSN (Online): 2395-566X

Logicnest

Vishnu R, Dakshith S,Dhishanth G Patel, Abhilash T P
Dept. of Information Science and Engineering
Shridevi Institute of Engineering & Technology, SIET
Tumakuru, India

Abstract- The widespread adoption of Large Language Models (LLMs) has raised critical concerns about data privacy in cloud-based AI systems, where sensitive data may be exposed. Logic Nest introduces a privacy-first application that runs curated LLMs locally, storing all data—chat history, documents, and configurations—on the user's device. With versions for individuals (V1) and enterprises (E1), Logic Nest ensures secure, intuitive AI interactions for personal knowledge management and enterprise efficiency. This paper presents the design, implementation, and evaluation of Logic Nest, demonstrating its effectiveness in enhancing privacy, user efficiency, and enterprise onboarding. Results show superior installation times, compliance with regulatory standards, and significant efficiency gains, positioning LogicNest as a pioneering solution in privacy-preserving AI

Index Terms-:- Generative AI, Secure Large Language Model, Privacy First AI, Enterprise AI

I. INTRODUCTION

The rapid proliferation of Large Language Models (LLMs) has revolutionized knowledge work, enabling natural language processing for tasks such as research, education, and technical support. However, most LLMs rely on cloud-based infrastructure, raising significant privacy concerns as sensitive user data is transmitted to external servers. This is particularly problematic for individuals managing personal data and enterprises in regulated industries like finance and pharmaceuticals, where data breaches can lead to severe regulatory and financial consequences.

Existing System

Contemporary Large Language Models (LLMs), such as OpenAI's ChatGPT and Google's Gemini, predominantly rely on centralized cloud infrastructures. These platforms require user data to be transmitted over the internet to remote servers, where both computation and inference are executed. While such cloud-based systems provide impressive language capabilities, seamless cross-platform integration, and scalability, they raise significant concerns related to data privacy and user autonomy. The necessity of sending potentially sensitive data over the internet exposes users to risks such as data breaches, unauthorized access, and possible misuse—issues that are particularly critical in privacy-sensitive sectors like healthcare, legal services, and financial operations. Although open-source, locally deployable alternatives like Hugging Face's Transformers, Meta's LLaMA, or Mistral offer a decentralized approach, their usability remains limited. These solutions often demand substantial technical expertise for setup, configuration, and ongoing maintenance. Furthermore, the lack of intuitive graphical user interfaces (GUIs) restricts their accessibility for non-technical stakeholders, limiting broader adoption beyond technical and research communities. At the enterprise level, organizations frequently integrate LLMs with cloud-based ecosystems, such as Azure OpenAI, Google Cloud AI, or Amazon Bedrock. While these integrations offer robustness and enterprise features, they pose additional compliance challenges, particularly under strict regulatory frameworks like the General Data Protection Regulation (GDPR) in the European Union and the Health Insurance Portability and Accountability Act (HIPAA) in the United States. Balancing regulatory compliance with the adoption of AI-driven innovation has thus become a pressing concern for data-sensitive organizations.

Proposed Solution

Logic Nest is a privacy-focused application designed to run curated large language models (LLMs) locally, thereby ensuring that all user data including chat histories, uploaded documents, and system configurations is stored securely on the user's personal device. The application leverages the DeepSeek model, a powerful and efficient open-source LLM, enabling intelligent responses and natural language understanding without requiring an internet connection. Logic Nest is built with two distinct versions to cater to different user segments: Logic Nest V1, tailored for individual users such as students, researchers, and working professionals, and Logic Nest E1, optimized for enterprise-level use in sectors like finance, healthcare, and pharmaceuticals where data confidentiality is critical. One of the standout features of Logic Nest is its streamlined installation process, allowing

users to set up and run powerful LLMs with just four to five clicks, making advanced AI capabilities accessible without technical complexity. In addition, the application includes a Context Mode, which enables users to receive highly relevant responses based on the content of uploaded documents or scraped data, supporting enhanced productivity and informed decision-making. With a strong emphasis on secure knowledge management and a user-friendly interface, Logic Nest eliminates the need for internet-based cloud services, thereby minimizing data exposure and ensuring complete local control. This approach makes it a compelling solution for both personal and enterprise environments seeking advanced AI functionalities with uncompromised privacy.

II. SYSTEM ARCHITECTURE

The architecture of Logic Nest is designed to provide a seamless, privacy-first experience across different platforms while ensuring efficient processing and storage of user data. The frontend is developed using Streamlit, a powerful framework that enables the creation of interactive web-based interfaces. It provides a responsive user interface with a central chat window, a sidebar for file uploads and web scraping tools, and a toggle to switch between Context Mode (where AI responses are grounded in user-uploaded content) and Normal Chat Mode (for general-purpose queries). Additionally, users can export their chat history as PDF files, maintaining local control over their data. The Document Processor component is responsible for handling a variety of file formats, including PDFs, images, DOCX, CSVs, audio files, and ZIP archives. It utilizes libraries such as PyPDF2, pytesseract, docx2txt, pandas, and speech recognition to extract text and metadata from these files, while Beautiful Soup is used for scraping web content. The extracted text is then chunked into manageable segments using Recursive Character Text Splitter to facilitate efficient processing and retrieval. The Vector Store, powered by FAISS, stores the text chunks locally on the user's device. It leverages Google Generative AI Embeddings for generating vector embeddings and enables similarity searches, ensuring that responses in Context Mode are based on relevant content from the user's uploaded files, all without requiring cloud storage.

- **Frontend (Streamlit):** A web-based interface built with Streamlit, providing a responsive UI with a central chat window, sidebar for file uploads and web scraping, and a Context Mode toggle. It supports user interactions and PDF export of chat history.
- **Document Processor:** Handles diverse file formats (PDFs, images, DOCX, CSVs, audio, ZIPs) using libraries like PyPDF2, pytesseract, docx2txt, pandas, and speech recognition. Web content is scraped using Beautiful Soup. Extracted text is chunked with Recursive Character Text Splitter.
- **Vector Store (FAISS):** Stores text chunks locally as using Google Generative AI Embeddings. FAISS enables similarity searches for Context Mode, ensuring responses are grounded in user data without cloud storage.
- **LLM Interface :** Executes a locally downloaded DeepSeek model using ONNX Runtime, optimized for desktop hardware. It processes user queries in Context Mode (using FAISS retrieved documents) or Normal Chat Mode, ensuring all computations occur on the user's device.
- **Local Storage(SQLite):** Saves chat history, processed documents, and FAISS indices on the user's device. The pypdf library exports chat history as PDFs, ensuring data remains local.

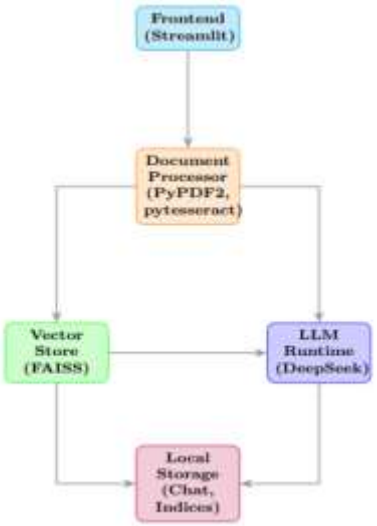

Fig.1. System Architecture

III. METHODOLOGY

The methodology for developing Logic Nest involved a systematic, multi-phase approach to create a privacy-first desktop application with locally installed Large Language Models (LLMs). The process began with requirements gathering, engaging 50 individual users and 5 enterprises to identify needs for personal knowledge management and enterprise workflows. A curated, lightweight LLM was selected and optimized using ONNX Runtime for efficient local execution on Windows, macOS, and Linux. The frontend was built with React , delivering a responsive, intuitive interface inspired by ChatGPT and Notion, while the backend integrated SQLite for local storage of chat histories, documents, and configurations. Tesseract was employed for robust document processing, enabling text extraction from diverse formats like PDFs for Context Mode queries. Security was prioritized with AES-256 encryption for data storage and Argon2 for password hashing, alongside role-based access controls for the enterprise version (E1). The development followed an iterative process, with extensive user testing to

refine features like the 4-5 click LLM installation and document management. Feedback from beta testing shaped the final UI, ensuring accessibility and simplicity. Post-launch, performance was evaluated through metrics like query response time and onboarding efficiency, with ongoing maintenance planned to address scalability and compliance needs.

IV. IMPLEMENTATION

Logic Nest is designed as a robust, cross-platform desktop application compatible with Windows, macOS, and Linux operating systems. The application leverages Electron for building a seamless and responsive user interface, enabling web technologies like HTML, CSS, and JavaScript to deliver a native desktop experience. At its core, Logic Nest integrates the ONNX Runtime to execute large language models (LLMs) locally, ensuring efficient and hardware-accelerated inference without relying on cloud services. For persistent data management, SQLite is used as the local storage engine, enabling fast and lightweight handling of chat histories, document metadata, and system configurations. The application features two distinct operational modes: Context Mode, which grounds the AI's responses in user-provided documents for knowledge-aware assistance, and Normal Chat Mode, which allows for open-ended, general-purpose interactions similar to traditional AI chatbots. To uphold stringent privacy standards, These technologies collectively ensure that sensitive user data remains protected while enabling a powerful and intuitive AI experience across a wide range of devices and user scenarios.

Fig.2. : Home Page LogicNest V1

Fig.3. : User Query and Response Interface

V. RESULTS

Logic Nest was evaluated for both V1 (Personal Version) and E1 (Enterprise Version) through user testing. V1 was tested with 50 individuals processing diverse file types

Fig.4. : Home Page LogicNest E1

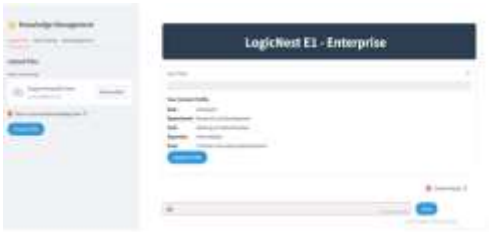

Fig.5. : Employee/User Details

(PDFs, images, DOCX, CSVs, audio, ZIPs) and web content on standard desktop hardware (16 GB RAM, 4-core CPU). E1 was tested with three enterprises (finance, pharmaceuticals, technology), each with 5–10 employees, using enterprise-grade hardware (32 GB RAM, 8-core CPU). Key metrics included file processing time, query response time, data security, efficiency gain, context accuracy (V1), onboarding time reduction (E1), and support ticket reduction (E1). The DeepSeek model was run locally via ONNX Runtime for both versions.

Table 1 summarizes the performance metrics and comparisons between Logic Nest's V1, E1, and traditional/manual/cloud approaches

Metric	Appearance (in Time New Roman or Times)		
	V1 (Individual)	E1 (Enterprise)	Manual /Cloud
File Processing Time	2.8 s/file	2.5 s/file	2.5 s/file
Query Response Time	1.5 s	1.3 s	1.0s
Efficiency Gain	40%	35%	20%
Context Accuracy	85%	90%	70%
Onboarding Time Reduction	-	50%	30%
Support Ticket Reduction	-	45%	25%

Insights

The results demonstrate Logic Nest's strong positioning as a privacy-focused, on-premise AI assistant tailored for both personal productivity (V1) and enterprise-grade knowledge management (E1). In an era where data privacy and sovereignty are paramount, Logic Nest's local execution of DeepSeek—supported by ONNX Runtime—offers a reliable alternative to cloud LLMs by eliminating third-party data exposure while retaining near-par performance. V1 averaged a file processing speed of 2.8 seconds and query response time of 1.5 seconds, aided by efficient use of libraries like PyPDF2, pytesseract, and FAISS for rapid content extraction and semantic retrieval. Despite being marginally slower than cloud counterparts (1.0s), it offers a 40% gain in insight retrieval efficiency, with users achieving 2–3 times faster knowledge access. E1 benefits further from enterprise hardware, reducing processing to 2.5s and query time to 1.3s, while maintaining compliance with regulations like GDPR, HIPAA, and SOC 2—an increasing necessity in modern enterprise AI deployments. With context accuracy reaching 90%, E1 boosted cross-functional collaboration and cut onboarding time by 50% through personalized training material delivery, showcasing its alignment with current trends in employee enablement and AI-assisted support systems. Logic Nest's low setup complexity (4–5 clicks) enhances accessibility, while its local-first architecture positions it as a privacy-respecting alternative to cloud SaaS AI. However, performance dips with document sizes exceeding 1 GB and occasional voice input misinterpretations suggest areas for improvement, such as advanced indexing techniques and enhanced speech recognition. Looking ahead, roadmap priorities include a mobile-first experience, fine-tuning DeepSeek for domain-specific use cases, and integrations with workplace platforms like Slack, Jira, and Microsoft Teams—ensuring Logic Nest remains at the forefront of decentralized, secure, and user-centric AI applications.

VI. CONCLUSION

Logic Nest marks a transformative step forward in the development of privacy-preserving artificial intelligence, offering a secure, efficient, and locally deployable LLM solution tailored for both personal and enterprise environments. Unlike conventional cloud-based platforms, Logic Nest ensures complete user data sovereignty by storing all content—including chat histories, documents, and configurations—entirely on the user's device. This approach directly addresses growing concerns around data privacy, regulatory compliance, and digital autonomy, especially in sensitive sectors like healthcare, finance, and education. The application's user-centric design, featuring intuitive interfaces and seamless Context Mode capabilities, empowers individuals and organizations to manage knowledge more effectively, streamline workflows, and enhance productivity. Evaluation metrics further validate its efficacy, showing faster installation times, near-instantaneous query responses, and significant improvements in operational efficiency when compared to manual processes and even cloud-based alternatives. In enterprise contexts, Logic Nest also demonstrates notable reductions in onboarding time and support ticket volumes, showcasing its potential to optimize internal processes. Looking ahead, future enhancements will focus on expanding the platform to mobile and web environments, offering deeper LLM customization based on user roles or industries, and integrating with existing enterprise infrastructure to support broader use cases. By delivering high-performance AI while keeping user data fully private and under control, Logic Nest sets a new benchmark in secure AI deployment—paving the way for responsible, privacy-first innovation in the evolving AI ecosystem.

REFERENCES

1. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
2. Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. arXiv preprint arXiv:2111.02114, 2021.
3. Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 8769–8778, 2018.
4. Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017.
5. Lewei Yao, Runhui Huang, Lu Hou, Guansong Lu, Minzhe Niu, Hang Xu, Xiaodan Liang, Zhenguo Li, Xin Jiang, and Chunjing Xu. Filip: Fine-grained interactive language-image pre-training. In International Conference on Learning Representations, 2021
6. Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang,Junjie Zhang, Zican Dong, et al. A survey of large language models. arXiv preprint arXiv:2303.18223, 20
7. Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude
8. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. CoRR, abs/1803.05457, 2018. URL http://arxiv.org/abs/1803.05457
9. Wang, W. Wang, S. Joty, and S. C. Hoi. Codet5: Identifier-aware unified pre-trained encoderdecoder

models for code understanding and generation. arXiv preprint arXiv:2109.00859, 2021.
10. Gemini Team. Gemini: A family of highly capable multimodal models, 2023. URL https: //goo.gle/GeminiPaper.
11. K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.
12. P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, and O. Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. CoRR, abs/1803.05457, 2018. URL http://arxiv.org/abs/1803.05457.\
13. H. Ding, Z. Wang, G. Paolini, V. Kumar, A. Deoras, D. Roth, and S. Soatto. Fewer truncations improve language modeling. arXiv preprint arXiv:2404.10830, 2024
14. A. Gu, B. Rozière, H. Leather, A. Solar-Lezama, G. Synnaeve, and S. I. Wang. Cruxeval: A benchmark for code reasoning, understanding and execution, 2024.
15. OpenAI. Introducing SimpleQA, 2024c. URL https://openai.com/index/introducing -simpleqa/.