iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** XI **Month of publication:** November 2025

DOI: https://doi.org/10.22214/ijraset.2025.74949

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

A Secure Pseudo-Hosted Framework for LLM Conversations in Virtual Environment

Krithiga B[1], Kavidha A[2], Marikkannan M[3]
[1]*M.E. Computer Science and Engineering (PG Scholar), Department of CSE, Government College of Engineering, Erode*
[2]*Associate Professor, Department of CSE, Government College of Engineering, Erode*
[3]*Assistant Professor (Senior), Department of CSE, Government College of Engineering, Erode*

Abstract: ***A privacy-preserving approach for interacting with Large Language Models (LLMs) using a pseudo-hosting methodology within a terminal-based environment. In today's world, most LLMs are accessed online through browsers or hosted cloud services, where user data such as prompts, responses, and metadata are often stored or tracked. This raises significant privacy concerns, including risks of data leaks, session hijacking, and unauthorized profiling. Meanwhile, local hosting offers privacy but demands powerful hardware, large memory allocation, and complex configurations. To bridge the gap between these two extremes, this paper proposes a pseudo-hosted LLM integration, where the model runs directly inside a terminal within a virtual environment (venv), ensuring isolation, data protection, and low system resource usage. The implementation uses Google's Gemini API key for establishing secure communication with the LLM, while dotenv is utilized to protect sensitive credentials by storing them in environment variables. The requests module facilitates query handling between the user and the model, ensuring smooth and efficient data exchange. For user convenience, the chatbot supports both text and voice-based interactions using speech_recognition for converting speech to text and pyttsx3 for converting text responses back into speech, providing a complete conversational experience through the command-line interface (CLI). Once the session ends, all chat data is automatically erased, ensuring 0% data retention and no history tracking, unlike typical online AI chat platforms. The pseudo-hosting methodology provides several advantages: it ensures enhanced privacy and data isolation, requires minimal hardware resources (no GPU, NPU, or large RAM), and functions entirely offline except for API requests. Additionally, by operating inside a virtual environment, it prevents any external scripts, cookies, or cached elements from accessing or recording user activity. Although it lacks a graphical interface and depends on API quotas, this approach successfully combines the privacy of local hosting with the ease and accessibility of online AI, forming a balanced, efficient, and secure LLM execution framework. The proposed system demonstrates that advanced LLMs can be integrated in a safe, lightweight, and privacy-conscious manner—marking a step forward in responsible AI usage and user data protection.***

Keywords: AI Chatbot, Data Privacy, Secure Communication, IDE Terminal, Session-Based Interaction, Local Execution, Privacy-Preserving System, Ephemeral Data, AI Ethics.

I. INTRODUCTION

Large Language Models (LLMs) have emerged as one of the most transformative innovations in modern computing, revolutionizing the way humans communicate with machines. From providing instant assistance and generating human-like responses to aiding research, software development, and education, LLM-based chatbots have seamlessly integrated into everyday digital workflows. Their accessibility through web-based platforms has further accelerated adoption, enabling users to interact with powerful language systems directly through browsers without any complex setup. However, this convenience also brings an inevitable concern—user privacy and data security. Most LLM chatbots available today operate on cloud infrastructures, where every user prompt and generated response is transmitted, processed, and often stored by third-party servers. These systems typically log interactions for performance optimization, analytics, or model fine-tuning. While such practices enhance language model accuracy and responsiveness, they simultaneously create potential vulnerabilities. Issues like session hijacking, unauthorized access, and the retention of sensitive data have raised serious privacy concerns among users. As LLMs become more capable of understanding personal or confidential information, ensuring data safety during interaction has become a crucial challenge.

One possible solution to these privacy risks is to deploy LLMs locally on personal devices. However, this method requires extensive computational resources—such as large amounts of RAM, VRAM, and GPU power—that are not feasible for average users. Consequently, there is a pressing need for a lightweight, privacy-conscious alternative that allows users to interact with LLMs without storing or exposing their data on the web.

This paper introduces a terminal-based LLM chatbot interface designed to provide privacy-preserving conversational experiences. The approach leverages the terminal environment inside an Integrated Development Environment (IDE), such as Visual Studio Code, which operates as an isolated execution shell. In this environment, no conversation history, cookies, or cached data are stored by default. Once the session ends, all previous interactions are permanently erased, ensuring that the conversation remains transient and inaccessible to external sources.

The proposed method acts as a balanced solution between full local hosting and traditional web-based usage. It enables users to benefit from the computational capabilities of hosted LLMs while maintaining session-level privacy through a terminal interface that leaves no digital footprint. This system emphasizes ephemeral data usage, secure interaction, and user control, presenting a practical and efficient response to the growing privacy challenges associated with LLM-driven communication. Moreover, as LLMs evolve to process increasingly complex and context-aware information, they inadvertently become repositories of user-generated content that may contain personal, academic, or professional details. When these models are hosted on centralized platforms, every interaction becomes part of a data cycle that is partially beyond the user's control. Even with anonymization and encryption, the potential for data retention or misuse remains a valid concern. Many users are unaware that their inputs can be temporarily or permanently stored, analyzed, and sometimes even used to train future iterations of the model. This lack of transparency in how conversational data is handled has intensified debates around ethical LLM usage and digital autonomy.

In contrast, the proposed terminal-based LLM interface creates a controlled and temporary environment for interaction. By integrating the chatbot within a system terminal, the process minimizes dependency on web browsers, which are typically embedded with trackers, cookies, and session managers. Terminals do not inherently support data caching or user activity tracking, which naturally limits the persistence of any communication. When the terminal is closed, the entire conversation instance disappears, thereby restoring user privacy instantly. This form of ephemeral communication ensures that no residual data remains accessible for external monitoring or exploitation. Furthermore, this approach offers a practical middle ground between fully offline and cloud-hosted systems. While local hosting ensures complete privacy, it is computationally demanding and often impractical for most users. On the other hand, web-based interfaces offer ease of access but at the cost of privacy and control. The terminal-based model, however, maintains connectivity with the LLM API while eliminating the storage and tracking mechanisms typically associated with browser environments. Thus, it combines the accessibility of online LLMs with the data protection benefits of local execution environments.

Beyond privacy, the system also contributes to the ethical and responsible deployment of LLMs. It aligns with emerging frameworks that prioritize user data rights, transparency, and minimal data persistence. This mode of operation ensures that users maintain ownership of their digital interactions while reducing the risks of unwanted profiling, targeted data collection, or information leakage. In academic, corporate, or research settings, where confidentiality is crucial, such a system could become a safe alternative for experimentation and day-to-day communication with LLMs. By focusing on transient interaction, non-persistence, and user autonomy, this paper emphasizes a growing need for lightweight, privacy-preserving methods of using language models. As LLMs continue to expand their influence across digital ecosystems, solutions that bridge the gap between usability and privacy will become indispensable. The proposed method aims to initiate a step toward that balance—enabling users to converse intelligently with machines while ensuring that their digital footprint remains entirely their own.

II. LITERATURE REVIEW

The concept of privacy has long been recognized as a cornerstone of individual freedom and democratic integrity. Although not explicitly stated in the U.S. Constitution, the right to privacy is deeply embedded within its legal and moral framework, serving as a safeguard for autonomy, liberty, and personal identity. Privacy creates essential zones for independent thought, self-expression, and emotional security, protecting individuals from undue intrusion and data exploitation. In the digital era, however, privacy has evolved beyond a moral principle—it has become a technical challenge. As online interactions continue to expand through web-based services and intelligent systems, protecting user data has become a central concern for both developers and policymakers [2].

The evolution of conversational systems dates back to the mid-20th century. Alan Turing introduced the concept that would define the trajectory of chatbot technology through his famous Turing Test, which questioned whether machines could exhibit human-like conversational behaviour. This foundational idea inspired the creation of ELIZA by Joseph Weizenbaum, which simulated a psychotherapist using a simple pattern-matching mechanism. Although primitive, ELIZA represented a breakthrough in natural language interaction and laid the groundwork for future research into machine understanding [4][6].

The following decades witnessed the emergence of more advanced systems such as PARRY, which incorporated emotion-based logic to simulate a patient with schizophrenia, and ALICE, which utilized Artificial Intelligence Markup Language (AIML) to enhance contextual responses. Despite their limitations, these systems demonstrated steady progress toward making computer dialogue more natural, leading to the eventual rise of modern AI-driven chatbots [7].

From the year onward, rapid advancements in computational power and data accessibility gave rise to sophisticated models built upon deep learning and natural language processing (NLP). The introduction of Large Language Models (LLMs), particularly OpenAI's GPT series and Google's Gemini, marked a paradigm shift in conversational intelligence. These models possess the ability to generate coherent, context-aware responses, making them integral to sectors like education, business, and healthcare. However, the same models that deliver exceptional linguistic capabilities also raise new challenges concerning data retention, transparency, and ethical governance [2][7].

Studies conducted in recent years have highlighted the trade-off between functionality and privacy in LLM applications. While these models improve their accuracy through continuous interaction logging and feedback analysis, this process inadvertently leads to the accumulation of user-generated data on centralized servers. Researchers have expressed growing concern about the implications of such practices, including risks of session hijacking, data profiling, and potential misuse of personal information. Despite existing data protection frameworks, ensuring privacy during real-time LLM interactions remains a largely unresolved issue in both academic and industrial contexts [13].

To address these challenges, recent studies have explored privacy-preserving techniques such as local deployment, federated learning, on-device inference, and temporary session storage. However, each of these methods presents trade-offs between accessibility, computational demand, and implementation complexity. Local hosting ensures maximum privacy but requires powerful hardware, while cloud-based solutions offer convenience but compromise user control. This research therefore situates itself within this ongoing discourse, proposing a terminal-based interface as a hybrid, lightweight alternative that preserves the transient nature of communication while maintaining accessibility [10].

In summary, the literature reveals a consistent evolution—from rule-based chatbots to context-aware LLMs—paralleled by an increasing awareness of privacy concerns. The proposed terminal-based model seeks to fill this research gap by offering a middle-ground approach that ensures user autonomy, minimizes data persistence, and aligns with the broader movement toward ethical, privacy-preserving LLM deployment [11].

III. EXISTING AND PROPOSED SYSTEM

A. Existing Systems

Existing deployment strategies for large language models (LLMs) are broadly categorized into local hosting and onsite/cloud-hosted systems, each with distinct benefits and limitations.

1) Local Hosting

Local hosting involves running the entire LLM directly on the user's hardware. While this method maximizes privacy and data control—since no queries leave the machine—it introduces significant technical challenges. LLMs are resource-intensive, requiring substantial RAM, VRAM, CPU/GPU capacity, and storage to load model weights and perform inference efficiently. Additionally, local hosting demands careful environment configuration, including dependency management, library installations, and permission handling. Even minor misconfigurations can result in errors or crashes, making local hosting accessible mainly to users with advanced technical knowledge. Furthermore, full local deployment limits portability and scalability, as running multiple models simultaneously or updating models can be cumbersome due to high memory consumption and prolonged load times.

2) Onsite / Cloud Hosting

Cloud-hosted LLMs provide a contrasting approach, allowing users to access powerful models without investing in high-spec hardware. This approach offers convenience, scalability, and rapid access to the latest model versions. However, it comes with significant privacy concerns. Conversation data and prompts are typically transmitted and stored on external servers, sometimes without transparent disclosure of data retention policies. This introduces potential vulnerabilities such as session hijacking, data leaks, or unauthorized analysis of sensitive queries, particularly in educational, research, or professional environments where confidentiality is critical. Moreover, cloud-based services may impose rate limits, access quotas, and tiered usage restrictions, which can limit continuous use or experimentation. In summary, local hosting prioritizes privacy but is highly resource-intensive and technically demanding, whereas cloud hosting prioritizes convenience and scalability but introduces privacy risks and potential data exposure. Both approaches represent trade-offs that may not meet the needs of users seeking secure, lightweight, and flexible access to LLMs.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

B. Proposed System

The pseudo-hosting methodology addresses the limitations of both local and cloud-hosted systems by combining their advantages while mitigating their drawbacks. In this approach, the LLM runs within a terminal-based interface encapsulated in a Python virtual environment (venv). The pseudo-hosted design ensures that all interactions remain ephemeral, with no persistent storage of prompts, responses, or session data. As a result, it effectively eliminates the risk of session hijacking, data leaks, and unauthorized access, while maintaining the convenience of remote API-based computation.

Unlike full local hosting, pseudo-hosting does not require large-scale hardware specifications. Since model computation occurs through the LLM API rather than on the local machine, memory, storage, and processing demands are minimal, making the system accessible on standard laptops or desktops. Users can engage in complex, multi-step interactions without investing in high-performance GPUs or NPUs.

Additionally, the virtual environment ensures dependency isolation, preventing conflicts with other software installed on the system. This encapsulation improves portability, reproducibility, and overall system stability, enabling users to maintain a controlled environment for experiments or research. The system supports text-based interaction, with optional voice input/output, and can be extended to future modalities without compromising privacy.

The pseudo-hosted model also facilitates flexible model selection within API-supported limits, allowing users to experiment with multiple LLMs and compare outputs in a secure, ephemeral setting. Model updates are handled through the provider (e.g., Gemini), which ensures that users benefit from improvements without the overhead of managing full model weights locally.

In essence, the proposed pseudo-hosted approach delivers a balanced, hybrid solution: it combines the privacy benefits of local hosting, the hardware efficiency and scalability of cloud hosting, and the simplicity of ephemeral, terminal-based interaction. This methodology establishes a new paradigm for secure LLM usage, particularly in educational, research, and professional environments where data confidentiality, lightweight execution, and usability are paramount.

```
    # Generate response using the Gemini model
        response = model.generate_content(user_input)
        chatbot_reply = clean_text(response.text)
        print("Chatbot:", chatbot_reply)
        speak(chatbot_reply)
```

IV. TECHNOLOGY USED

The development of the terminal-based pseudo-hosted LLM interface leverages a combination of external and internal Python modules to ensure smooth, lightweight, and privacy-preserving functionality.

A. External Modules

- google.generativeai – Provides access to the Gemini LLM, enabling natural language processing and response generation within the terminal.
  (Ref: import google.generativeai as genai)
- dotenv – Facilitates secure access to sensitive environment variables, including API keys, without hardcoding credentials.
- requests – Handles HTTP requests to the LLM API, allowing queries and receiving responses programmatically.
- speech_recognition – Converts spoken input into text, supporting voice-enabled interactions within the pseudo-hosted terminal interface.
  (Ref: import speech_recognition as sr)
- pyttsx3 – Provides offline text-to-speech functionality, allowing responses to be audibly delivered without relying on cloud services. (Ref: import pyttsx3)

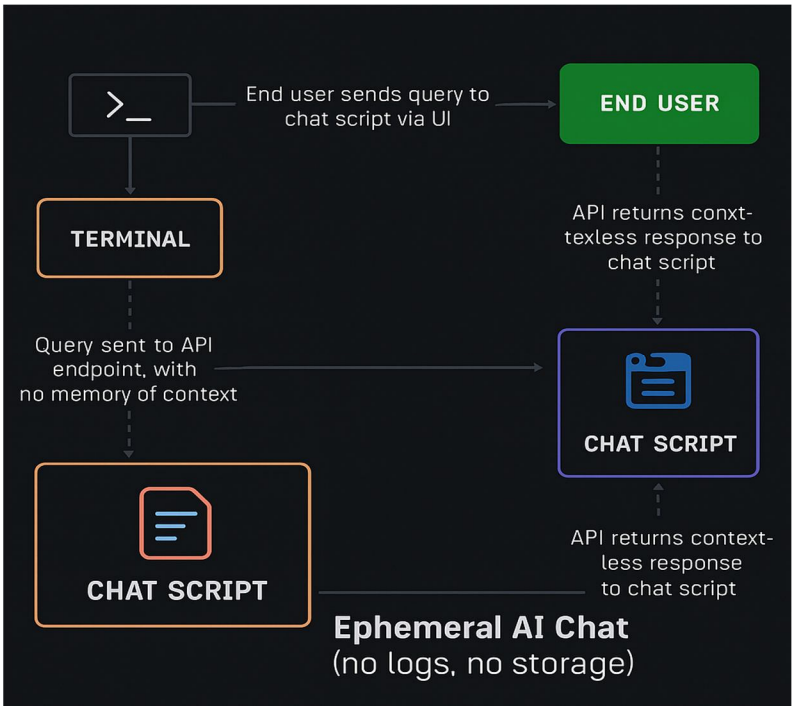
B. Internal Modules

- re – Utilized for regular expression matching to parse and interpret user input efficiently.
- os – Provides access to system environments and file paths, essential for configuring virtual environments and secure handling of temporary data.

This combination of modules allows the interface to operate with minimal hardware requirements, maintain isolated virtual environments, and provide both text and optional voice interaction while preserving privacy.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

V. SYSTEM ARCHITECTURE



(Fig.1) Workflow Architecture

Fig.1 represents the architecture of an ephemeral AI chat system, designed to prioritize privacy by eliminating all forms of data persistence. At its core, the system allows an end user to interact with an AI chatbot directly through a terminal interface, instead of a web-based UI. The terminal serves as a minimal and local communication layer, receiving user input and displaying the AI's responses without storing any chat logs or history. Inside this terminal runs a chat script—a lightweight local program, typically written in Python or JavaScript—that manages the flow of communication between the user and the remote AI API. When the user enters a query, the chat script sends it as a single, stateless request to the API endpoint, meaning no previous conversation data or context is included. The AI API, hosted by a model provider (like OpenAI or Anthropic), processes the request, generates a context-free response, and sends it back to the chat script. The script then displays the response in the terminal and discards it afterward. Throughout this process, no local or remote logs are maintained beyond the temporary memory of the running process, ensuring complete ephemerality. Once the terminal is closed, all conversation data vanishes, leaving no trace on disk or cloud services. This design achieves what can be called pseudo-hosting—a method where the user runs the AI interaction client locally, without hosting or storing any persistent data themselves. The result is a secure, stateless, privacy-centric system ideal for sensitive use cases where confidentiality and minimal data exposure are paramount.

VI. APPLICATIONS

A. Privacy-Sensitive Academic Work

In educational contexts, students and researchers can leverage the terminal-based LLM for tutoring, coding assistance, and idea exploration without exposing sensitive data to cloud servers. Queries, assignments, or draft research inputs exist only within the ephemeral session and are erased upon closing the terminal. This ensures that confidential academic content remains entirely under the user's control, enabling experimentation and learning in a zero-data-retention environment.

B. Secure Professional Productivity

Professionals working with proprietary data, confidential documents, or sensitive communications can use the terminal-based system as a private assistant. The pseudo-hosting setup allows the LLM to process tasks such as text summarization, email drafting, or code generation without creating server-side logs or traces, eliminating potential vectors for session hijacking. Teams can safely perform exploratory tasks in real-time, knowing that all interactions are isolated and ephemeral.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue XI Nov 2025- Available at www.ijraset.com

C. Research and Development Testing

For AI researchers and developers, the system provides a safe sandbox to test prompts, experiment with model outputs, or evaluate LLM responses without persistent storage. By simulating a local-hosting experience without the computational overhead of full offline models, the interface allows rapid iteration while ensuring 100% data privacy. This pseudo-hosting methodology makes it ideal for sensitive R&D projects where information leakage could have serious consequences.

D. Ethical and Privacy-First Deployments

The terminal-based approach exemplifies a privacy-by-design philosophy, making it suitable for organizations that require compliance with stringent data protection policies. Its ephemeral sessions guarantee that interactions leave no trace, enabling ethical deployment of LLMs in environments where confidentiality, user autonomy, and data security are paramount.

E. Low Hardware Requirements

Unlike traditional local hosting solutions that require high RAM, VRAM, storage, or specialized neural processing units (NPUs), the pseudo-hosted terminal interface operates with minimal hardware demands. Since the LLM runs within a lightweight execution environment and relies on API-based computation rather than full local inference, users can access full conversational capabilities without overloading system resources. This ensures accessibility even on standard laptops or desktops, making it practical for a wide user base.

F. Virtual Environment Isolation

The entire LLM and its supporting libraries are encapsulated within a Python virtual environment (venv). This ensures that the model and its dependencies are isolated from the host system, preventing conflicts with other software and adding an extra layer of security. By keeping the model contained within a virtual environment, the system enhances portability, reproducibility, and safety while further reducing the risk of unintentional data leakage or system-wide exposure.

VII. LIMITATIONS

A. Command-Line Interface Only

The system is limited to a command-line interface (CLI), which, while lightweight and secure, offers no graphical user interface (GUI). The absence of a GUI may present usability challenges for users unfamiliar with terminal operations or those who rely on visual workflows for complex interactions. For example, tasks involving multi-step prompts, text formatting, or structured input may be cumbersome without visual cues.

B. API Usage Constraints

The system relies on external LLM APIs, such as Gemini, for model computation. As a result, it is subject to API usage quotas and restrictions, similar to free-tier online LLM services. Users may encounter limitations in the number of requests, response sizes, or session frequency, which can impact productivity during intensive use.

C. Limited Model Availability

Currently, the terminal-based interface supports only a subset of LLMs provided through the Gemini API. Users cannot freely switch to alternative models or self-hosted LLMs without significant configuration and computational resources.

D. No Live Updates or On-the-Fly Training

The interface does not support dynamic updates or incremental learning of the LLM. Any improvements, bug fixes, or new capabilities must be applied externally through updates from the Gemini API. This limitation restricts the adaptability of the system, preventing it from evolving in real-time with new data or user feedback. For use cases that require continuous learning or immediate integration of domain-specific knowledge.

E. Dependency on External Service Maintenance

The pseudo-hosted system is inherently dependent on the availability and reliability of the Gemini API. Any downtime, maintenance, or policy changes implemented by the service provider can temporarily disable functionality or necessitate reconfiguration. This reliance on an external provider limits the autonomy of the pseudo-hosted approach, making operational continuity contingent on external factors beyond the user's control.

F. Minimal Offline Capabilities

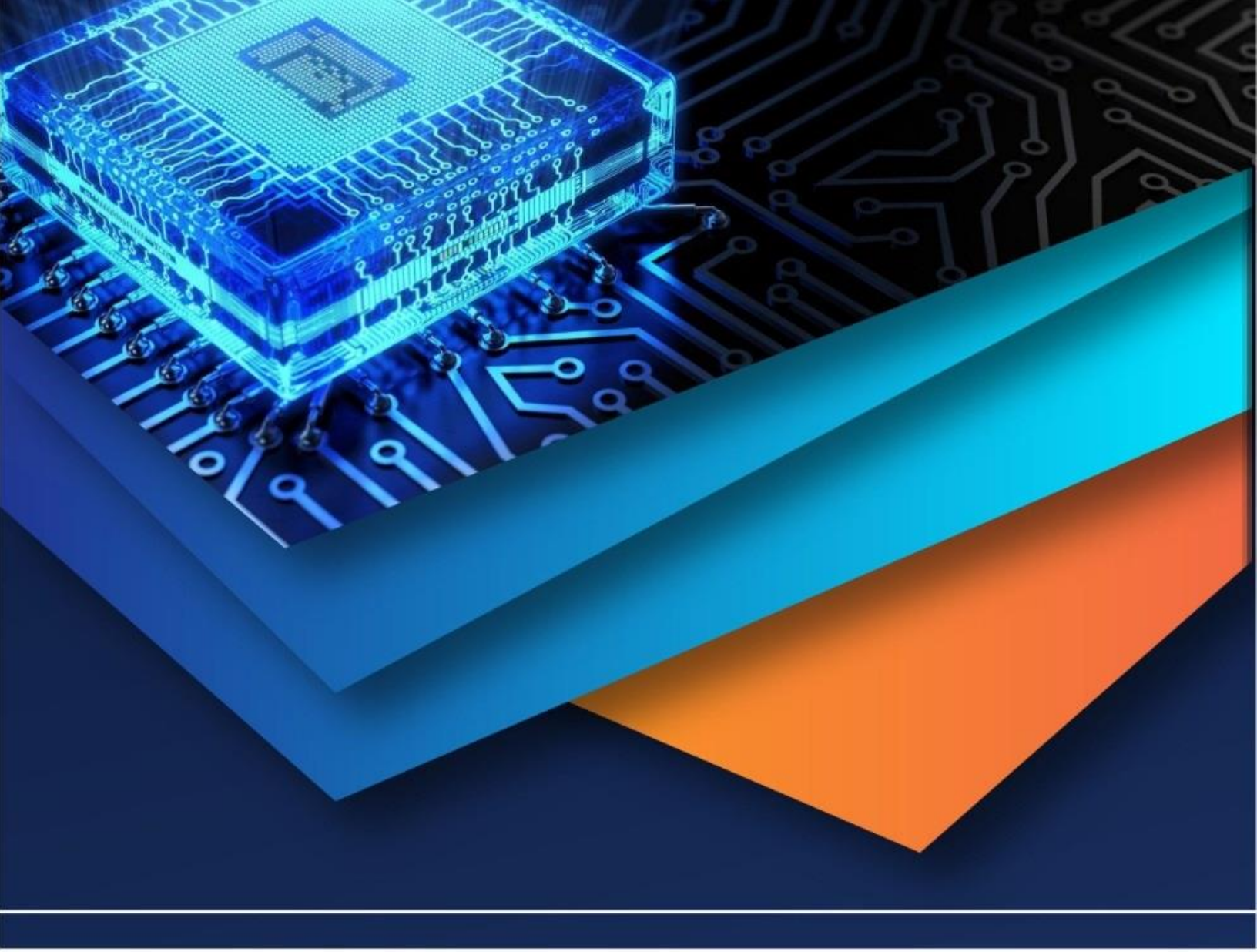
While the pseudo-hosted approach reduces hardware requirements and ensures privacy, it does not fully operate offline. Unlike fully local deployments, computation still requires an active connection to the LLM API, making it unsuitable for environments with intermittent or restricted internet access.

VIII. CONCLUSION AND FUTURE ENHANCEMENT

The implementation of the pseudo-hosted terminal-based LLM interface demonstrates a significant advancement in balancing usability, privacy, and accessibility for everyday users. The vast majority of people leverage AI to complete tasks efficiently and effectively, with seven out of ten individuals using AI in their daily routines, yet only a small fraction of users actively considers privacy concerns. Traditional AI platforms often lack transparency regarding user data processing, highlighting a critical gap between convenience and data security. The proposed approach addresses this gap by running the LLM within a virtual or isolated environment, akin to VS Code's Shell and virtual environment variables, allowing the model to function fully while ensuring that user data remains local, ephemeral, and secure. This design allows users to interact with AI without exposing their conversations to potential session hijacks or unauthorized storage, though it comes with minor trade-offs, such as the absence of a graphical user interface and the need for some manual configuration. Despite these compromises, users still gain full functionality and output quality comparable to cloud-hosted AI, but with complete control over privacy. Furthermore, the pseudo-hosting methodology opens avenues for future enhancements, including the integration of diffusion models or custom server-based deployment, which would allow even more sophisticated AI interactions while maintaining privacy and offering UI customization. While more computationally intensive models like diffusion models may require external GPUs or additional VRAM for optimal performance, the current system demonstrates that a secure, low-spec, terminal-based LLM can serve as a practical, privacy-conscious alternative for a wide range of users, bridging the gap between convenience, efficiency, and data security.

REFERENCES

[1] M. Kahng et al., "LLM Comparator: Interactive Analysis of Side-by-Side Evaluation of Large Language Models," in IEEE Transactions on Visualization and Computer Graphics, vol. 31, no. 1, pp. 503-513, Jan. 2025.
[2] D. S. Johnson, O. Hakobyan, J. Paletschek and H. Drimalla, "Explainable AI for Audio and Visual Affective Computing: A Scoping Review," in IEEE Transactions on Affective Computing, vol. 16, no. 2, pp. 518-536, April-June 2025.
[3] J. Berengueres, "How to Regulate Large Language Models for Responsible AI," in IEEE Transactions on Technology and Society, vol. 5, no. 2, pp. 191-197, June 2024.
[4] L. Benaddi, C. Ouaddi, A. Jakimi and B. Ouchao, "A Systematic Review of Chatbots: Classification, Development, and Their Impact on Tourism," pp. 78799-78810, vol. 12, in IEEE Access, 2024.
[5] D. Noori Zadeh and M. B. Elamien, "Generative AI for Analog Integrated Circuit Design: Methodologies and Applications," pp. 58043-58059, vol. 13, in IEEE Access, 2025.
[6] S. Iftikhar, S. H. Alsamhi and S. Davy, "Enhancing Sustainability in LLM Training: Leveraging Federated Learning and Parameter-Efficient Fine-Tuning," in IEEE Transactions on Sustainable Computing,2024.
[7] S. Karnouskos, "The Relevance of Large Language Models for Project Management," pp. 758-768, vol. 5, in IEEE Open Journal of the Industrial Electronics Society, 2024.
[8] Q. Lu, L. Zhu, X. Xu, Z. Xing and J. Whittle, "Toward Responsible AI in the Era of Generative AI: A Reference Architecture for Designing Foundation Model-Based Systems," pp. 91-100, vol. 41, no.6, in IEEE Software, 2024.
[9] T. Scantamburlo et al., "Software Systems Compliance with the AI Act: Lessons Learned from an International Challenge," pp. 44-51, in IEEE/ACM International Workshop on Responsible AI Engineering (RAIE), Lisbon, Portugal, 2024.
[10] A. Koubaa, W. Boulila, L. Ghouti, A. Alzahem and S. Latif, "Exploring ChatGPT Capabilities and Limitations: A Survey," pp. 118698-118721, vol. 11, in IEEE Access, 2023.
[11] O. Mubin, F. Alnajjar, Z. Trabelsi, L. Ali, M. M. A. Parambil and Z. Zou, "Tracking ChatGPT Research: Insights from the Literature and the Web," pp. 30518-30532, vol. 12, in IEEE Access, 2024.
[12] A. T. Neumann, Y. Yin, S. Sowe, S. Decker and M. Jarke, "An LLM-Driven Chatbot in Higher Education for Databases and Information Systems," pp. 103-116, vol. 68, no. 1, in IEEE Transactions on Education, 2025.
[13] P. Haindl and G. Weinberger, "Students Experiences of Using ChatGPT in an Undergraduate Programming Course," pp. 43519-43529, vol. 12, in IEEE Access, 2024.
[14] S. Yu, F. Carroll and B. L. Bentley, "Insights Into Privacy Protection Research in AI," pp. 41704-41726, vol. 12, in IEEE Access, 2024.
[15] M. Amin Kuhail, M. Bahja, O. Al-Shamaileh, J. Thomas, A. Alkazemi and J. Negreiros, "Assessing the Impact of Chatbot-Human Personality Congruence on User Behavior: A Chatbot-Based Advising System Case," pp. 71761-71782, vol. 12, in IEEE Access, 2024.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY