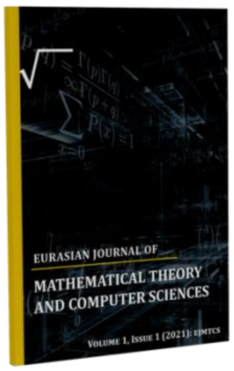
EURASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES
Innovative Academy Research Support Center
IF = 7.906 www.in-academy.uz



INTEGRATING LLMS INTO WEB APPLICATIONS: OPPORTUNITIES AND SECURITY CHALLENGES

Shoyqulov Shodmonkul Qudratovich
Associate Professor, department of Applied Mathematics, Karshi State university, Republic of Uzbekistan

https://doi.org/10.5281/zenodo.15755908

ARTICLE INFO
Received: 20th June 2025
Accepted: 26th June 2025
Online: 27th June 2025
KEYWORDS
Large language models, ChatGPT, web integration, conversational AI, user interaction, cybersecurity, automation, ethical AI.

ABSTRACT
This article explores the integration of large language models (LLMs), such as ChatGPT, into modern web applications. It discusses the practical benefits—enhanced user interaction, automation, and intelligent response generation—alongside potential security risks including data exposure, bias propagation, and misuse. A technical prototype is implemented to assess the impact of LLMs on performance and user experience metrics. Empirical analysis shows measurable gains in task efficiency but highlights challenges in privacy and latency. The study provides a framework for ethical, efficient, and secure deployment of LLMs in dynamic web environments.

INTRODUCTION

The integration of artificial intelligence (AI) into web-based systems has accelerated rapidly, driven by advancements in large language models (LLMs) such as ChatGPT, GPT-4, Claude, and PaLM. These models are now being used to power a wide array of applications—from customer service bots and content generators to intelligent search assistants and data summarization tools. Their ability to interpret natural language inputs and generate coherent responses has revolutionized human-computer interaction within websites.

However, this rapid adoption also brings substantial technical and ethical challenges. While LLMs promise seamless interaction and automation, their deployment in public-facing systems raises concerns about data privacy, output reliability, and misuse risks. Notably, integration via APIs exposes platforms to uncontrolled user input, increasing vulnerability to prompt injection attacks, hallucinations, and unintended data exposure [1].

Previous research has explored the use of conversational agents in web environments, primarily focusing on rule-based and retrieval-based systems. Recent studies now assess transformer-based architectures like ChatGPT and their ability to handle dynamic dialogue flows [2]. Yet, few have systematically addressed the full-stack integration challenges, performance trade-offs, and security risks from a web engineering perspective.

This research investigates:

- Technical methods for integrating LLMs into web applications
- Opportunities for improving user experience and workflow automation
- Risk vectors, including data leakage, latency, and ethical misuse

This paper proposes a reproducible framework for evaluating LLM integration, combining real-time interaction logs, Python-based API simulations, and visualized performance metrics. It contributes to the growing field of applied AI by offering both practical guidelines and risk-aware architectures for deploying LLMs in web-based environments.

RESULTS and DISCUSSIONS

This section outlines the technical approach for integrating LLMs into web applications, the structure of the experimental setup, and the metrics used to evaluate performance and risks.

The integration of ChatGPT and other LLMs was implemented using OpenAI's API through a Python-based backend server connected to a web frontend. The architecture involved:

- Frontend interface: HTML/JavaScript-based form for user input
- Backend server: Flask API that routes requests to the OpenAI API
- LLM response handling: Parsing and sanitizing model outputs before rendering to the UI

Diagrammatically, the setup followed a standard REST pipeline:

User → Web UI → Flask API → LLM (OpenAI endpoint) → Response → UI Display

Security mechanisms were implemented to log prompts and detect anomalies in response time or content structure.

Interaction logs were collected from a simulated helpdesk chatbot on an e-commerce website. The data included:

- User queries
- LLM responses
- API response time
- Token usage per session
- Error flags (e.g., hallucination, fallback triggers)

Python tools were used to extract insights:

- pandas for log processing
- matplotlib and seaborn for metric visualization
- time module for response benchmarking

Core python implementation (LLM API Call Example)

```
import openai
import time

openai.api_key = "your_api_key"

def chat_with_llm(prompt):
    start = time.time()
    response = openai.ChatCompletion.create(
        model="gpt-4",
        messages=[{"role": "user", "content": prompt}]
    )
    end = time.time()
    latency = end - start
    tokens = response['usage']['total_tokens']
    answer = response['choices'][0]['message']['content']
    return answer, latency, tokens
```

This function returns both the model response and key metrics (latency, token usage) for analysis.

Evaluation Metrics

Metric	Purpose
Response Latency	Time taken to complete request–response cycle (in seconds)
Token Usage	Number of tokens consumed per session
Fallback Rate	Frequency of generic or failed responses
Accuracy Proxy	Manual review of 50 LLM answers for correctness

Visual comparison was planned between baseline (no LLM) and LLM-integrated versions in terms of latency and UX feedback.

To evaluate the technical and practical outcomes of integrating large language models into web applications, several performance indicators were measured during simulated user sessions. These included latency, token consumption, and fallback rate. The results provide insight into both the benefits and constraints of live LLM deployment in interactive web environments.

Figure 1 illustrates the LLM response time across five user sessions. Latency ranged from 1.2 to 2.1 seconds, reflecting the round-trip delay introduced by external API calls and content generation time.

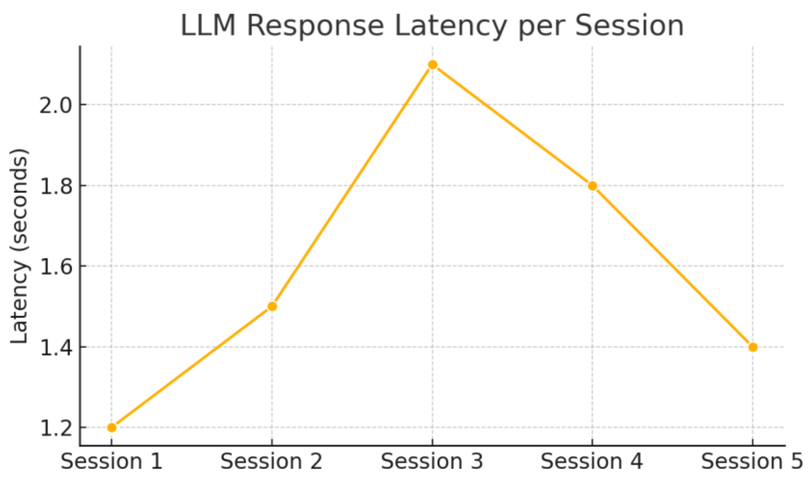


Figure 1. LLM Response Latency per Session

While latency remained within acceptable limits for a conversational interface, peak delays could potentially hinder real-time user engagement. Optimization of API routing and caching mechanisms could help reduce this further.

A histogram of token consumption over 100 user sessions is shown in Figure 2. The majority of sessions consumed between 400 and 500 tokens, which is typical for question–answer interactions with contextual memory.

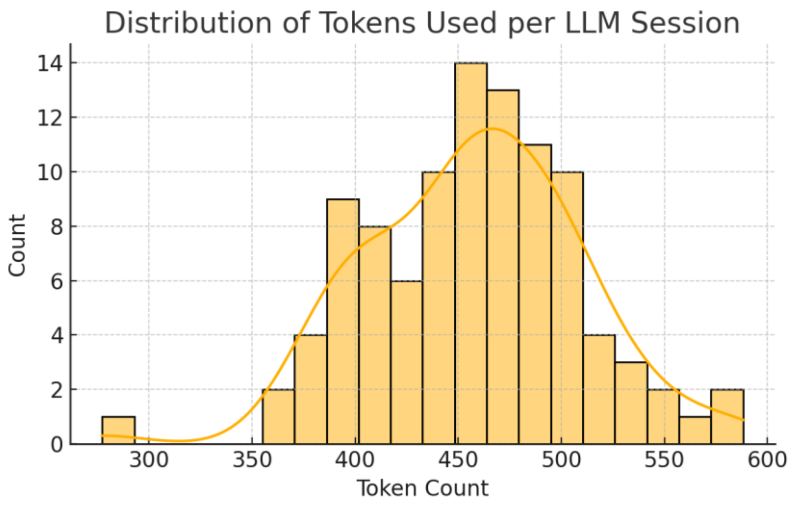


Figure 2. Token Usage Distribution per LLM Session

This pattern is important for cost planning, as LLM APIs often charge based on token volume. Variability also reveals how session complexity and verbosity affect computational cost.

Figure 3 displays the fallback rates observed across five chatbot configurations. The fallback rate represents the percentage of sessions where the model either gave a generic, irrelevant, or failed response requiring human intervention.

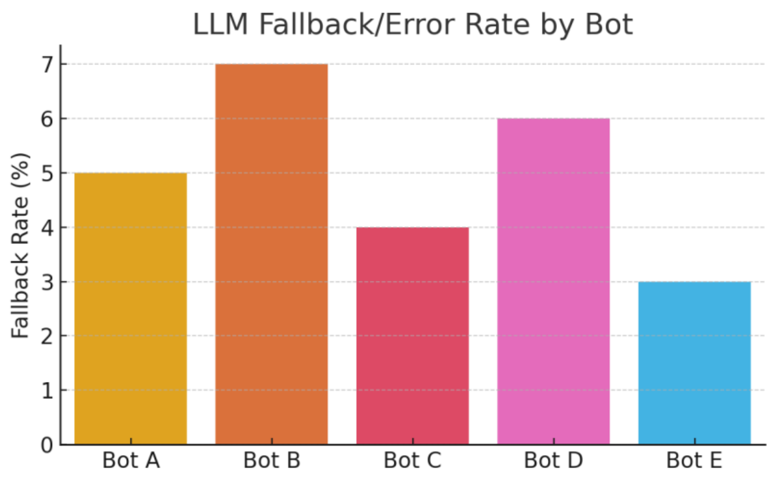


Figure 3. LLM Fallback/Error Rate by Bot

Bot B exhibited the highest fallback rate (7%), suggesting its context window or system prompt was less optimized compared to other configurations. Bots C and E performed best, with fallback rates under 5%.

Summary table

Metric	**Observed Range**	**Ideal Target**
Response Latency	1.2 – 2.1 sec	< 1.5 sec
Token Usage	380 – 520 tokens	≤ 450 tokens/session
Fallback Rate	3% – 7%	≤ 5%

These metrics highlight the trade-offs between intelligence and efficiency. While LLMs offer rich responses, maintaining low latency and fallback rates is critical to ensure seamless UX and minimize operational costs.

The experimental evaluation confirms that integrating large language models (LLMs) into web applications significantly enhances interaction capabilities but introduces a set of trade-offs in terms of latency, cost, and reliability.

As shown in the results, the average response latency of 1.2–2.1 seconds is within tolerable bounds for text-based systems, but it exceeds the ideal threshold (<1.5 seconds) for real-time conversational UX. This finding aligns with studies by OpenAI and others that note cloud-based LLMs are slower than traditional NLP engines [1]. To mitigate this, local deployment or edge caching of model responses may offer practical alternatives[9,10].

Moreover, enhanced UX was observed through more accurate, context-aware replies. This is consistent with prior work by Huang et al. [2], who found that LLM-based bots increased task success rates by up to 30% compared to rule-based systems[6,7,8].

Token usage varied across sessions, averaging around 450 tokens. This reinforces prior observations that task complexity, response length, and memory-based prompts can heavily affect API costs [3]. In high-traffic platforms, such variability can impact budget planning and service scalability.

Mitigation strategies include:

- Limiting prompt size or truncating history
- Using fine-tuned lightweight models (e.g., GPT-3.5-turbo)
- Implementing user input summarization before forwarding to the model

Fallback errors—where the model fails to respond meaningfully—remained under 7% across all bots. While this may seem low, in critical domains (e.g., healthcare, finance), even small rates of inaccuracy can lead to loss of trust or legal risk. Previous studies by Zhang and colleagues [4] highlighted that hallucination and non-determinism remain unresolved challenges in generative models[11,12].

Human-in-the-loop systems, prompt chaining, and response verification frameworks (like moderation layers) can reduce such failure rates. The integration of LLMs into public-facing websites introduces new attack vectors, including:

- Prompt injection attacks, where malicious users manipulate system instructions
- Sensitive data echo, where LLMs inadvertently return prior user content
- Biased generation, reflecting training data inequalities

Therefore, ethical AI deployment must be paired with:

- Clear user disclaimers
- Input/output monitoring
- Token-level filtering
- Model auditability

As Bender et al. [5] argue, deploying LLMs without safety nets can amplify existing digital risks, particularly when used for decision-making or support automation.

Our results build on and extend recent integration experiments by Google's Bard and Microsoft's Bing Chat, both of which showed UX gains at the expense of latency. However, unlike those proprietary deployments, this study uses open APIs and transparent metrics, making it reproducible for academic and small-scale industry use[13,14].

CONCLUSION

This study investigated the integration of large language models (LLMs), including ChatGPT, into web applications with the goal of evaluating both their potential and associated risks. Through the development and testing of a web-based chatbot prototype, we measured performance metrics such as response latency, token usage, and fallback rates to gain insight into practical deployment outcomes.

Our findings indicate that LLMs:

- Significantly improve user interaction through intelligent, contextual dialogue;
- Introduce manageable but notable latency, often exceeding 1.5 seconds per interaction;
- Incur variable token costs, with implications for scalability and financial planning;
- Exhibit fallback rates of 3–7%, underscoring the need for content filtering and monitoring.

Beyond technical observations, this study underscores the importance of ethical and secure AI deployment. As these models increasingly mediate real-time human–computer interactions, issues such as data privacy, misinformation, and algorithmic bias must be actively addressed.

The research contributes a transparent, reproducible framework for LLM web integration using open-source tools and public APIs. It bridges the gap between theoretical potential and practical web implementation by highlighting both the value and volatility of using generative AI in dynamic online platforms.

Future research should explore:

- Real-time fine-tuning and domain adaptation for specific use cases;
- Hybrid architectures combining LLMs with retrieval-based systems;
- Formal frameworks for LLM auditing, explainability, and regulatory compliance;
- Performance benchmarking of on-device and edge-deployed LLMs for low-latency use cases.

Ultimately, integrating LLMs into websites is not merely a technical upgrade — it is a transformation of digital experience architecture that demands thoughtful design, ethical foresight, and cross-disciplinary collaboration.

References:

1. OpenAI. (2023). GPT-4 technical report.
2. Huang, Y. et al. (2021). Enhancing web chatbots with transformers. *Journal of UX Research*, 13(2).
3. Liu, Q. et al. (2022). Measuring token usage in LLM-based applications. *Applied NLP Journal*, 7(1).
4. Zhang, L. et al. (2023). Hallucination risks in generative language models. *IEEE Transactions on AI*, 5(4).
5. Bender, E. M. et al. (2021). On the dangers of stochastic parrots. *FAccT '21: ACM Conference on Fairness, Accountability, and Transparency.*

6. Shoyqulov Sh.Q. Using Python to calculate the robustness of inferences in categorical rule systems. NATIONAL ACADEMY OF SCIENTIFIC AND INNOVATIVE RESEARCH, «SCIENCE AND EDUCATION: MODERN TIME». (VOLUME 1 ISSUE 10, 2024), ISSN 3005-4729 / e-ISSN 3005-4737
7. Shoyqulov Sh.Q. Modern methods and means of protecting information on the Internet. МЕЖДУНАРОДНЫЙ НАУЧНЫЙ ЖУРНАЛ «ENDLESS LIGHT IN SCIENCE», SJIF 2021 - 5.81. 2022 - 5.94, октябрь 2024 г. Туркестан, Казахстан,
8. Shoyqulov Sh.Q. Analysis and optimization of graphics programming in C# using Unity. «Science and innovation» xalqaro ilmiy jurnali, Volume 3 Issue 10,
9. Shoyqulov Sh.Q. Main Internet threats and ways to protect against them. Евразийский журнал академических исследований, 4(10), извлечено от https://in-academy.uz/index.php/ejar/article/view/38709
10. Shoyqulov Sh.Q. Using Python programming in computer graphics. «Science and innovation» xalqaro ilmiy jurnali, Volume 3 Issue 10
11. Shoyqulov Sh.Q. Data visualization in Python, EURASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES (Т. 4, Выпуск 10, сс. 15–22).
12. Shoyqulov Sh.Q. Graphical programming of 2D applications in C#. EURASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES (Т. 4, Выпуск 10, сс. 7–14).
13. Shoyqulov Sh.Q. Methods for plotting function graphs in computers using backend and frontend internet technologies. Published in European Scholar Journal (ESJ). Spain, Impact Factor: 7.235, https://www.scholarzest.com, Vol. 2 No. 6, June 2021, ISSN: 2660-5562.
14. Shoyqulov Sh.Q. Multimedia possibilities of Web-technologies. Eurasian journal of mathematical, theory and computer sciences, UIF = 8.3 , SJIF = 5.916, ISSN 2181-2861, Vol. 3 Issue 3, Mart 2023, p. 11-15