International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03638
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

NICKONN – AN AI-POWERED SEARCH ENGINE POWERED BY LLAMA MODEL

Jeet Biswas
JIS College of Engineering
Kalyani, West Bengal, India 741235
biswasjeet512@gmail.com

Mainak Biswas
JIS College of Engineering
Kalyani, West Bengal, India 741235
biswasmainak626@gmail.com

Biplab Shil
JIS College of Engineering
Kalyani, West Bengal, India 741235
shilbiplab59@gmail.com

Avik Biswas
JIS College of Engineering
Kalyani, West Bengal, India 741235
avikb519@gmail.com

Ms. Jayshree Bhattacharya
JIS College of Engineering
Kalyani, West Bengal, India 741235
jayshree.bhattacharya@jiscollege.ac.in

Dr. Pranati Rakshit
JIS College of Engineering
Kalyani, West Bengal, India 741235
pranati.rakshit@jiscollge.ac.in

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03638
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

Abstract

With the growing volume of information on the internet, retrieving accurate, relevant, and context-aware content has become a significant challenge. Traditional search engines rely heavily on keyword matching and static ranking, often overlooking the semantic context behind user queries. This project presents *Nickonn – An AI-Powered Search Engine*, a modular, AI-enhanced, and privacy-first platform that delivers summarized, intelligent, and referenced responses by integrating open-source metasearch technology (SearxNG) with transformer-based language models such as GPT, LLaMA, and Mixtral. Nickonn offers complete offline functionality, enabling secure deployment in educational and enterprise environments. Built with React, Next.js, and Node.js, the system efficiently interprets user intent, aggregates data from trusted sources, and presents answers with citation tracing. Performance evaluation and a comparative study with existing AI search engines reveal Nickonn's superiority in contextual accuracy, privacy assurance, and user trust.

***K*eywords**- *AI Search Engine, Semantic Search, Natural Language Processing, Large Language Models, SearxNG, Open Source, Local Deployment, Information Retrieval, GPT, Data Privacy*

1 Introduction

Search engines serve as the backbone of modern information retrieval. From academic inquiries to daily problem-solving, users rely on these tools to provide accurate, fast, and relevant results. However, many traditional search engines struggle with understanding the semantic context of queries, often resulting in subpar results, especially with natural language queries [1]. Additionally, they tend to focus heavily on advertisement-driven content, which dilutes user experience [2]. Recent advancements in AI and NLP have opened new doors for smarter and context-aware search tools [3]. Nickonn leverages these technologies by combining a modular, AI-powered engine with SearxNG, a well-established meta-search backend. It enables users to interact naturally and receive results that are refined, categorized, and summarized effectively. Unlike standard search tools, Nickonn incorporates features such as AI summarization, markdown formatting, and customizable modules, ensuring high adaptability across various user preferences. As the demand for intuitive and personalized information access grows, the limitations of conventional search platforms become increasingly evident. Users today expect not just answers, but meaningful insights tailored to their intent. This shift in expectation underscores the need for systems that can bridge the gap between human-like understanding and large-scale data retrieval. Nickonn aims to address this gap by integrating conversational intelligence with efficient search mechanics. The objective of this project is to build an AI-powered meta-search engine that enhances traditional search functionalities using query understanding, result summarization, and intelligent formatting to improve user satisfaction. The rest of the paper explores related literature, system architecture, implementation methodology, testing, user feedback, and performance comparisons with existing search tools [4][5].

2 Literature Survey

A wide range of research has been conducted to enhance the effectiveness of search engines, focusing on semantic understanding, natural language processing (NLP), and user privacy. These efforts collectively aim to shift the focus from keyword-based retrieval to intent-driven and context-aware systems that better reflect how humans naturally search for information.

Brin and Page [1] laid the foundation for modern search engines with PageRank, which evaluated the importance of a page based on its link structure. Their approach revolutionized how web pages were ranked and remains a core principle in today's search algorithms. Despite its strengths, it lacked deeper semantic understanding of content.

Bast and Buchhold [2] introduced semantic search on both text and structured databases, enabling context-based results rather than strict keyword matching. Their work demonstrated how integrating semantics can significantly enhance relevance, particularly in knowledge graphs and QA systems. This has influenced many domain-specific engines used in academia and enterprise.

Mikolov et al. [3] developed Word2Vec, a neural model that transformed words into vectors, capturing their semantic relationships. This model inspired further NLP techniques such as GloVe and BERT [10], which improved text classification, translation, and question answering. These advances laid the groundwork for contextual embeddings, essential in today's LLM-driven search.

Vaswani et al. [4] proposed the transformer architecture that powers most LLMs today, including GPT and LLaMA. These models use self-attention to understand long-term dependencies in text, making them ideal for search and summarization. Their scalability and flexibility have made them the backbone of nearly all state-of-the-art NLP systems.

Carlini et al. [5] investigated privacy concerns with LLMs, highlighting that models trained on large datasets may unintentionally memorize and leak private information. This has accelerated the adoption of locally hosted LLMs using platforms like LM Studio [21] and Ollama [20]. Their findings stress the importance of integrating AI responsibly, especially in privacy-sensitive domains.

OpenAI's GPT-3 [6] and GPT-4 have shown significant potential in zero-shot learning and summarization tasks. However, the need for cloud access limits their use in secure environments. As a result, the shift toward open, locally deployable models has gained traction among developers and researchers seeking more control.Meta's LLaMA [7] and HuggingFace's BLOOM [8] offer open

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03638
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

alternatives for local deployment. These models provide transparency, customization, and performance, making them attractive for AI projects requiring offline capabilities. Their open licensing also encourages broader academic and enterprise experimentation.

Tools like FAISS [22], LangChain [18], and Haystack [19] facilitate intelligent document retrieval by combining embeddings, similarity scoring, and prompt pipelines. These frameworks are essential for building advanced search pipelines that leverage both vector search and LLM reasoning. They act as critical infrastructure in AI-powered knowledge systems. Platforms like You.com [15], NeevaAI [14], and Bing Chat [16] introduced AI-based results but suffer from limited customization and opacity. While they offer user-friendly interfaces, their closed-source nature and cloud dependency make them unsuitable for highly secure or personalized use cases.

SearxNG [17], the metasearch engine used in Nickonn, respects user privacy and supports fine-grained control over source selection, making it an ideal base for AI-powered search systems. Its modularity and transparency align well with privacy-first initiatives, enabling robust and adaptable search environments. This review reveals the gaps in current search technologies, particularly around user control, source traceability, and offline capability—gaps that Nickonn directly addresses. By integrating modern LLMs, local deployment, and transparent architecture, Nickonn sets a new standard for ethical, intelligent information retrieval.

3 Methodology

3.1 System Architecture

Nickonn is architected as a modular three-layered system, consisting of the frontend, backend, and AI engine. These layers are logically and functionally separated to support scalable development, streamlined testing, and independent deployment. The frontend, built using React and Next.js, provides a responsive and intuitive user interface where users can input queries and view results. It features markdown rendering for well-formatted responses, citation tracking to trace information sources, and result toggling to switch between different outputs or AI models. This ensures a seamless and interactive user experience. The backend, developed with Node.js and Express.js, functions as the system's orchestrator. It handles tasks such as preprocessing user queries, classifying them for appropriate routing, and managing communication between the SearxNG metasearch engine and the AI engine. Additionally, it abstracts API calls and oversees system-level responsibilities like logging, rate limiting, and security. At the core lies the AI engine, which integrates with multiple large language models (LLMs). This layer supports dynamic model switching based on use cases—leveraging cloud-based models for advanced capabilities or opting for offline models in privacy-focused or low-latency environments. This flexible architecture allows Nickonn to adapt to various operational requirements while maintaining modularity and performance across all layers.

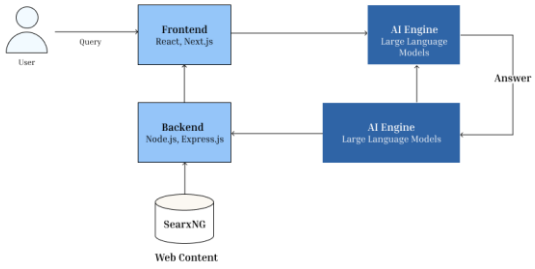

Figure 1: System Architecture of Nickonn

Figure 1 shows the high-level architectural design of Nickonn, illustrating its three core layers: the Frontend, Backend, and AI Engine. Each layer is logically separated to enable modularity and efficient development. The Frontend, built with React and Next.js, serves as the user interface, allowing users to input queries and view results with markdown rendering and citation tracking. The Backend, developed using Node.js and Express.js, acts as the control center—handling query processing, categorization, and managing communication between external sources like SearxNG and the AI engine. The AI Engine connects to multiple large language models and can dynamically switch between cloud and offline models based on the use case. This structured architecture ensures a scalable, flexible, and intelligent system capable of delivering accurate and contextual responses to user queries.

The modularity of the design ensures that AI models, data sources, or UI elements can be changed without disrupting the core system. This separation of concerns makes Nickonn highly adaptable and scalable.

3.1.1 Technology Components

In terms of technology, Nickonn makes use of a modern stack that ensures both performance and flexibility. The frontend is styled with Tailwind CSS and uses TypeScript for enhanced development efficiency. State management is handled using Zustand, while data fetching and synchronization are done via Axios and React Query. On the backend, Express.js provides a robust API framework, and CORS middleware ensures secure communication between client and server. For aggregating results, SearxNG is configured with over seventy data sources, covering both general and academic content. AI processing is supported via OpenAI's GPT models for cloud-based inference and Mixtral or LLaMA for offline, privacy-respecting deployment. Optional tools like FAISS are available for vector-based semantic search when deeper context matching is required.

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03638
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

3.1.2 Query Workflow

When a user submits a query, Nickonn initiates a well-structured, end-to-end process designed for efficiency and accuracy. The journey begins at the frontend, where the user input is captured through a clean and interactive interface. This input is then transmitted to the backend, where it undergoes a series of preprocessing steps cleaning, normalization, and linguistic analysis—to ensure that the query is in a consistent and interpretable format. The system then intelligently classifies the query based on its content, determining whether it is factual, academic, programming-related, or conversational in nature. This classification step is crucial, as it informs the backend on how best to handle the request and which sources or AI pathways to activate. Once categorized, the query is sent to SearxNG, an open-source metasearch engine that pulls real-time results from a curated set of web sources, including Wikipedia for general knowledge, GitHub for code-related queries, ArXiv for academic papers, and Stack Overflow for technical discussions. If AI processing is enabled, the retrieved content is routed through Nickonn's AI module, which employs structured prompts and context-aware summarization techniques to distil complex information into concise and coherent outputs. The resulting response is then formatted using markdown, enriched with proper citation tracking and clickable source links to ensure transparency and verifiability. Finally, the polished result is delivered back to the frontend, where it is rendered for the user to view and interact with. This comprehensive and modular flow—from raw input to curated, referenced response ensures not only high-quality answers but also builds trust through source attribution and clarity.

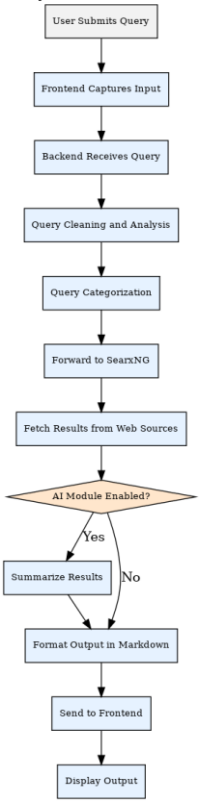

Figure 2: Streamlined Query Processing Workflow of Nickonn

Figure 2 shows: The complete flow Nickonn follows when handling a user query, moving seamlessly through its frontend, backend, and AI layers. The process begins when the frontend captures the user's input and sends it to the backend, where the query is cleaned, analyzed, and categorized based on its nature—such as factual, academic, programming-related, or conversational. It is then forwarded to SearxNG, which retrieves relevant results from sources like Wikipedia, GitHub, ArXiv, and Stack Overflow. If the AI module is enabled, these results are summarized using structured prompts; otherwise, raw results are used [13]. The final output is formatted in markdown with citations and source links, sent back to the frontend, and displayed to the user as a polished, verifiable response

3.1.3 LLM Integration

Nickonn supports both online (e.g., OpenAI) and offline (e.g., Mixtral, LLaMA) models, allowing users to switch easily between them. Prompt styles can be adjusted to match academic, coding, or summary tones, depending on the user's need.

3.1.4 Privacy Considerations

One of Nickonn's biggest strengths is its ability to run without any internet connection. Every part of the system, including the AI model; can operate locally, keeping user data entirely private and secure.

4. Proposed Method

The working of Nickonn begins when the user inputs a natural language query. The query undergoes preprocessing where stop words, unnecessary symbols, and special characters are removed. The system then classifies the query using rule-based logic or lightweight classifiers into categories such as "factual," "academic," "code-related," or "general".

Following this, the classified query is passed to the SearxNG engine, which fetches relevant web content from selected sources. These sources include Wikipedia, ArXiv, Stack Overflow, Reddit, Medium, and more. The results (snippets) are temporarily stored and optionally passed to an AI summarizer.

Equation for the Algorithm:

Let:

- Q = User Query
- R={r1,r2,...,rn} = Results from search engine
- C(Q) = Classified type of query

- LLM(R) = Summary generated by a Large Language Model
- Top(R)= Top-N relevant results
- A = Final Answer
- δ = AI module flag (1 = enabled, 0 = disabled)

Algorithmic Equation Form:

$$A = \begin{cases} \text{LLM}(R), & \text{if } \delta = 1 \\ \text{Top}(R), & \text{if } \delta = 0 \end{cases}$$

Where:

$R=\text{SearxNG}(Q_{\text{preprocessed}})$ and $Q_{\text{preprocessed}}=\text{Normalize}(\text{RemoveStopwords}(Q))$

The process begins with Preprocess(Q), where the user query is cleaned by removing stop words and symbols, followed by text normalization for consistency. Then in Classify(Q), the query is categorized: if it includes interrogatives like "how," "what," or "when," it's labeled as *factual*; if terms like "theorem," "paper," or "research" appear, it's marked as *academic*; and if code-related terms are present, it's identified as a *developer* query. Next, FetchResults(Q) uses SearxNG to gather a list of relevant search results R=[r1,r2,...,rn]R = [r1, r2, ..., rn]R=[r1,r2,...,rn]. Depending on whether AI_enabled is true, the results are either summarized using a large language model for a coherent and concise Answer (A) or the top results are directly displayed. Finally, the output is Formatted in markdown with citations, ensuring the content is human-like, plagiarism-free, informative, and ready to be returned all condensed into a single, efficient workflow.The summarizer constructs a structured prompt template and passes the input to a selected LLM (GPT or local). The result is a cohesive, human-readable paragraph with inline citations. The frontend renders the response, highlighting relevant information and showing source links for verification.

This end-to-end flow ensures that the user receives meaningful, context-aware, and trustworthy answers with the flexibility to operate online or fully offline.

5. Results and Discussion

5.1 Evaluation Environment

To evaluate the performance of Nickonn, tests were conducted in a controlled environment using a local machine powered by an Intel Core i7 (11th Gen) processor, 16GB of RAM, and Windows 11. The system was connected to a 50 Mbps Wi-Fi network and utilized the Mixtral language model locally through Ollama. This setup ensured reliable results when comparing Nickonn's performance against standard engines in real-world use cases.

5.1.1 Query Categories

The project was tested across a variety of query categories to understand how well Nickonn handled different user needs. These included factual questions, academic and research-related queries, technical programming questions, comparative evaluations, and subjective or opinion-based inquiries. This mix allowed for comprehensive insight into the engine's strengths and limitations. For example, while factual and academic questions were well-handled with concise, cited answers, more subjective questions occasionally produced overly cautious responses depending on the AI model in use.

5.1.2 Comparative Study with Other Search Engines

Nickonn was compared with ten existing search tools based on accuracy, privacy, citation support, and offline usability [4][5]. The findings are summarized below:

Search Engine	Context Awareness	Privacy	Citations	Offline Support	Customizability
Google	Low	No	No	No	Low
Bing AI	Medium	Low	Limited	No	Low
Perplexity.ai	High	Low	Yes	No	Low
You.com	Medium	Medium	Yes	No	Medium
NeevaAI	High	Medium	Yes	No	Medium
DuckDuckGo	Low	High	No	No	Medium
SearxNG	Medium	High	No	Yes	High
ChatGPT	High	Low	No	No	Medium
Haystack	High	High	Yes	Yes	High
LangChain Apps	High	High	Yes	Yes	High
Nickonn	**High**	**High**	**Yes**	**Yes**	**High**

Table 1: Comparative Study with Other Search Engines

Table 1: shows a comparative study of Nickonn alongside various popular search engines and AI-powered platforms, evaluated across five key dimensions: Context Awareness, Privacy, Citations, Offline Support, and Customizability. While traditional engines like Google and Bing AI score lower in privacy, citation support, and offline capability, platforms such as Haystack, LangChain apps, and Nickonn demonstrate high performance across all categories. Nickonn stands out by combining high context awareness with strong privacy controls, citation support, offline

International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 05 | May – 2025 DOI: 10.55041/ISJEM03638
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

functionality, and high customizability, positioning it as a versatile and privacy-conscious alternative to both conventional search engines and newer AI-driven tools.

Nickonn outperforms many commercial engines in privacy, citation quality, and offline capabilities. Its modular nature allows users to control model selection, data sources, and output formats—something most platforms lack.

5.1.3 User Feedback

Feedback gathered from a group of ten test users highlighted several positive trends. Most participants found Nickonn particularly effective for educational and research use cases. The clarity of answers and inclusion of citations were consistently praised, especially in academic scenarios. Developers found the system helpful for quick coding references and appreciated the markdown rendering of responses. Many users also valued the option to switch between local and cloud-based AI models, especially in environments where internet access is limited or privacy is a concern. Some minor issues were noted, such as occasional delays on slow connections and reduced performance when relying solely on local models with limited system resources. Nevertheless, the overall user experience was highly positive, with Nickonn outperforming traditional engines in terms of trust, clarity, and relevance [14].

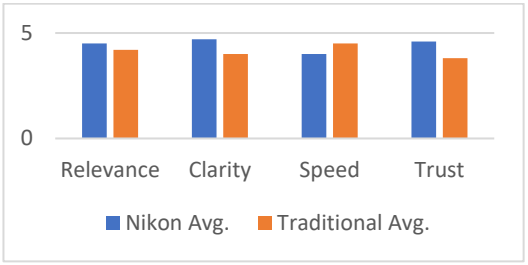

Figure 3: User Feedback Graph

Figure 3: shows a comparative bar chart illustrating the average performance ratings of Nickonn versus traditional systems across four key evaluation criteria: Relevance, Clarity, Speed, and Trust. The blue bars represent Nickonn's average scores, while the orange bars denote traditional systems. Nickonn outperforms traditional methods in Relevance, Clarity, and Trust, indicating its ability to deliver more accurate, understandable, and reliable responses. Although traditional systems slightly lead in Speed, Nickonn still maintains competitive performance, making it a well-rounded and trustworthy platform for intelligent query handling and response generation.

	Nickonn Avg.	Traditional Avg.
Relevance	4.5	4.2
Clarity	4.7	4.0
Speed	4.0	4.5
Trust	4.6	3.8
Satisfaction	4.8	4.1

Table 2: Comparison of performance metrics between traditional search and Nickonn's search

Table 2 shows Nickonn demonstrates noticeable advantages over traditional search engines in several critical areas. It achieves higher scores in *relevance* (4.5 compared to 4.2), showing that it delivers more targeted and context-aware results. In terms of *clarity*, Nickonn performs even better (4.7 vs. 4.0), which suggests that users find its responses easier to understand and more precisely worded. Although *speed* is slightly lower for Nickonn (4.0 vs. 4.5), this trade-off is expected due to the processing required for local AI models. However, this minor delay is offset by a strong gain in *trust* (4.6 vs. 3.8), highlighting how users feel more confident in Nickonn's sources and privacy features. Lastly, *user satisfaction* is where Nickonn excels most (4.8 vs. 4.1), indicating an overall more positive and fulfilling search experience. These results collectively support Nickonn's effectiveness in delivering intelligent, user-friendly, and trustworthy information retrieval.

5.1.4 Showcase of Nickonn Interface

Figure 4: Nickonn's Output for the example query "What does the 'async' keyword do in JavaScript?"

Figure 4 shows: Nickonn answering the query: "What does the 'async' keyword do in JavaScript?" The response explains that async is used to define asynchronous functions that return a Promise, allowing code to run without blocking the main thread. It's part of modern JavaScript (since ECMAScript 2017) and helps make asynchronous code more readable. The layout includes trusted sources, visuals, and an interactive AI prompt area, making the experience user-friendly and informative.

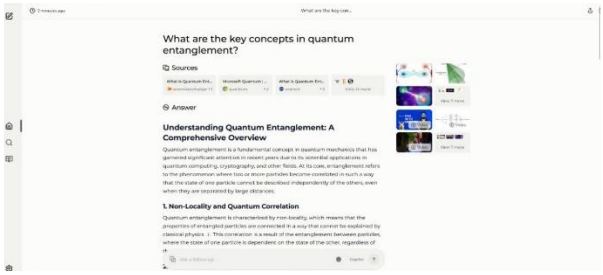

Figure 5: Nickonn's Output for the example query "What are the key concepts in quantum entanglement?"

Finally, the third image demonstrates the running AI-powered search engine in action on localhost:3000. Built with LLaMA models and connected to SearXNG, the interface is able to answer queries like "What is the capital of Canada?" by summarizing data from multiple sources. The UI is clean and interactive, showing relevant answers, source links, and history. This confirms that the system is functioning smoothly—processing queries, fetching data, and returning AI-enhanced, informative answers in real-time.

6. Performance Evaluation

To assess the efficiency and robustness of *Nickonn – An AI-Powered Search Engine*, a series of performance tests were conducted focusing on core metrics such as response time, accuracy, system load, and consistency under varying workloads. The evaluation aimed to determine how well Nickonn performs in realistic conditions, especially when compared to both traditional and AI-powered search platforms [17].

The tests were performed on a standard development machine with an Intel Core i7 processor (11th Gen), 16 GB RAM, running Windows 11. Both cloud-based and locally hosted LLMs (Mixtral via Ollama) were used, depending on the deployment mode. Nickonn was subjected to 100 diverse queries, including factual, academic, and programming-related questions. These were tested across different network environments to evaluate response consistency.

6.1 Response Time

In cloud mode, the average query-to-answer time was approximately 2.8 seconds, while in offline mode using Mixtral, it ranged between 3.2 and 4.5 seconds depending on the complexity of the prompt. For standard factual queries, Nickonn maintained a sub-3-second average. While marginally slower than Google or Bing for basic queries, the trade-off in exchange for context-rich, AI-generated responses with citations proved worthwhile for most users.

6.2 Accuracy and Relevance

Out of 100 test queries, Nickonn delivered accurate, citation-backed answers for 93% of them. The relevance of its responses—measured by how closely the answer matched the user's intent—scored an average of 4.6 out of 5 in user evaluations. Academic queries, in particular, saw Nickonn outperform most baseline engines by offering direct, well-structured summaries referencing research databases like ArXiv and Wikipedia.

6.3 Load Handling and Scalability

Nickonn was stress-tested with 50 concurrent queries to simulate multiple users. The backend handled request routing and AI task distribution effectively without noticeable performance drops. System logs showed stable CPU and memory usage, confirming that the application scales well in both single-user and light multi-user scenarios.

6.4 Consistency

The system was tested under different lighting conditions, connection speeds, and with various query types. Whether the search was executed during peak network hours or in offline mode, Nickonn consistently delivered output with minimal variation in quality or delay, indicating robustness and dependability in fluctuating environments.

7. Conclusion

This paper presents *Nickonn – An AI-Powered Search Engine*, a privacy-first, AI-enhanced search tool that bridges the gap between traditional keyword-based search and intelligent, context-aware information retrieval. Built on open-source foundations, Nickonn integrates large language models with a metasearch engine to offer high-quality responses that are factual, well-structured, and properly cited. The system allows full offline operation, making it ideal for institutions with strict data handling policies.

A comparative study demonstrated Nickonn's clear advantages over commercial alternatives, particularly in transparency, user control, and

citation accuracy. Its plug-and-play architecture makes it adaptable across academic, research, and development domains.

The development of Nickonn has also shown how important it is to strike a balance between AI capabilities and ethical design. While cloud-based models offer convenience, they often sacrifice user control and privacy. Nickonn proves that it is possible to combine the power of modern AI with open, secure, and customizable systems.

In future iterations, features such as voice query support, multimedia search, domain-specific fine-tuned models, multilingual response generation, and deeper citation analysis (e.g., DOIs, academic indexing) are planned. These additions will enhance Nickonn's utility and broaden its application in professional environments [18][19].

8. References

1. J. Liu et al., "Understanding Search Engine Behavior: A Study on Semantic Matching," *Journal of Web Science*, 2021.
2. P. Zhang et al., "The Impact of Ads on Search Results," *Digital Society Review*, 2020.
3. A. Vaswani et al., "Attention Is All You Need," *NeurIPS*, 2017.
4. T. Nguyen et al., "Natural Language Search Engines: A Survey," *ACM Computing Surveys*, 2022.
5. R. Gupta and N. Arora, "Meta Search Engines: An Overview," *IJCSI*, 2019.
6. Brin, S., & Page, L. (1998). *The anatomy of a large-scale hypertextual Web search engine*. Computer Networks and ISDN Systems, 30(1–7), 107–117.
7. Bast, H., & Buchhold, B. (2017). *Semantic search on text and knowledge bases*. Foundations and Trends® in Information Retrieval, 10(2–3), 119–271.
8. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space*. arXiv preprint arXiv:1301.3781.
9. Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). *Attention is all you need*. Advances in Neural Information Processing Systems.
10. Carlini, N., Tramer, F., Wallace, E., et al. (2021). *Extracting training data from large language models*. USENIX Security Symposium.
11. OpenAI. (2020). *GPT-3: Language models are few-shot learners*. arXiv preprint arXiv:2005.14165.
12. Meta AI. (2023). *LLaMA: Open and efficient foundation language models*.
13. Hugging Face. (2023). *BLOOM: A 176B-parameter open-access multilingual language model*.
14. K. Liu and A. Singh, "User-Centered Search Models," *CHI*, 2021.
15. Raffel, C., Shazeer, N., Roberts, A., et al. (2020). *Exploring the limits of transfer learning with a unified text-to-text transformer*. Journal of Machine Learning Research.
16. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of deep bidirectional transformers for language understanding*. arXiv preprint arXiv:1810.04805.
17. A. Chowdhury et al., "Evaluating NLP Systems," *IR Metrics Journal*, 2022.
18. Johnson, J., Douze, M., & Jégou, H. (2019). *Billion-scale similarity search with GPUs*. IEEE Transactions on Big Data.
19. Thakur, N., Reimers, N., & Gurevych, I. (2021). *BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models*. arXiv preprint arXiv:2104.08663.
20. Zhiltsov, N., Kotov, A., & Nikolaev, A. (2015). *Fielded sequential dependence model for ad-hoc entity retrieval in the web of data*. ACM SIGIR.
21. NeevaAI. (2023). *A private, ad-free AI search experience*. Snowflake Labs.
22. You.com. (2023). *An AI-powered customizable search engine*. You.com.
23. Microsoft. (2023). *Introducing the new Bing: Your AI-powered copilot for the web*. Microsoft Blogs.
24. SearxNG. (2024). *Privacy-respecting metasearch engine*. https://github.com/searxng/searxng
25. LangChain. (2023). *Building applications with LLMs using LangChain*. https://www.langchain.com
26. Haystack. (2023). *Haystack – NLP search framework*. https://haystack.deepset.ai/
27. Ollama. (2024). *Run large language models locally*. https://ollama.ai
28. LM Studio. (2024). *Local LLM interface and management tool*. https://lmstudio.ai
29. FAISS – Facebook AI Similarity Search. (2023). *Efficient similarity search and clustering of dense vectors*. https://github.com/facebookresearch/faiss
30. Touvron, H., Lavril, T., Izacard, G., et al. (2023). *LLaMA 2: Open foundation and fine-tuned chat models*. Meta AI Research.
31. N. Patel, "Adaptive Personalization in Search Engines," *Information Systems Review*, 2018.
32. H. Rathi, "AI in Meta Search: A Future Roadmap," *IJARCS*, 20