International Journal of Computer Applications (0975 – 8887)
Volume 187 – No.93, March 2026

System Design and Evaluation of a Tab Management Extension

Rutuja Habib
Department of Computer Engineering
Anantrao Pawar College of Engineering & Research
Pune, Maharashtra, India

Sahil Sarode
Department of Computer Engineering
Anantrao Pawar College of Engineering & Research
Pune, Maharashtra, India

Zaid Tamboli
Department of Computer Engineering
Anantrao Pawar College of Engineering & Research
Pune, Maharashtra, India

Medha Sapkal
Department of Computer Engineering
Anantrao Pawar College of Engineering & Research
Pune, Maharashtra, India

Jitendra Musale, PhD
Department of Computer Engineering
Anantrao Pawar College of Engineering & Research
Pune, Maharashtra, India

ABSTRACT
Managing a large number of active browser tabs often leads to excessive memory consumption, reduced responsiveness, and degraded user experience, especially during multitasking-intensive workflows. This paper presents the design, implementation, and deployment of the Adaptive User-Centric Tab Management System (AUCTMS), a browser extension that performs context-aware tab grouping and hybrid priority scoring based on access recency, usage frequency, semantic relevance, and memory footprint. The system is evaluated under realistic multitasking scenarios with 30–50 concurrent tabs, reflecting typical user browsing behavior in research, development, and content consumption. Experimental results demonstrate significant reductions in memory usage, improved browser responsiveness, and minimal tab restoration latency. These findings confirm that behavior-aware, context-driven tab management can be effectively deployed as a lightweight, client-side extension, providing practical improvements in multitasking efficiency and system stability.

General Terms
Browser Extensions, Memory Management, Resource Optimization, Web Performance

Keywords
Browser extension implementation, user-centric tab management, context-aware tab grouping, hybrid priority scoring, adaptive tab suspension, runtime memory optimization, browser performance optimization

1. INTRODUCTION
Modern web browsers have evolved into complex platforms supporting multitasking activities such as research, software development, online collaboration, and content creation. Users frequently open multiple tabs simultaneously, which increases memory consumption, reduces responsiveness, and creates frequent context-switching overhead. These issues are particularly noticeable on resource-constrained systems, where inefficient tab utilization can degrade user experience.Existing tab management techniques primarily rely on static heuristics such as least-recently-used policies, manual tab grouping, or simple suspension rules. Some research explores intelligent tab clustering and predictive tab behavior; however, many approaches remain conceptual, computationally intensive, or unsuitable for real-time execution within browser extensions. This highlights a clear gap between theoretical tab management models and practical, deployable solutions.

To address this gap, this paper presents the **Adaptive User-Centric Tab Management System (AUCTMS)**, a fully implemented browser extension that performs context-aware tab grouping and hybrid priority scoring based on access recency, usage frequency, semantic relevance, and memory footprint. The system is lightweight, operates entirely on the client side, and adapts dynamically to user behavior and memory constraints while preserving session state for suspended tabs.

The primary contributions of this work are:

1. **Design and implementation of a deployable browser extension** capable of real-time tab analysis and optimization without external computation.
2. **Adaptive suspension mechanism** that balances memory reduction with fast tab restoration.
3. **Experimental evaluation under realistic browsing workloads** involving 30–50 concurrent tabs, demonstrating significant memory savings, improved browser responsiveness, and minimal tab restoration latency.

This implementation-driven approach bridges the gap between conceptual models and practical deployment, validating that behavior-aware, context-driven optimization can enhance system stability and multitasking efficiency in modern web browsers.

2. LITERATURE REVIEW
Memory and resource management in computing systems has been a widely studied area. Lee and Park [1] proposed an efficient memory management technique for mobile operating systems based on relaunch distance prediction, demonstrating improvements in system responsiveness. Similarly, Lei et al. [5] explored domain-based memory isolation for WebAssembly applications, showing that careful allocation can prevent memory conflicts and enhance stability. Ali et al. [6] presented a context-aware model for dynamic working memory updates, highlighting the importance of adaptive memory strategies for varying workloads. These studies

emphasize predictive and adaptive memory management but are mostly focused at the OS or runtime level rather than in-browser tab behavior.

In the domain of web browsing, tab management has emerged as a key challenge. Lin and Washburn [8] developed a Chrome extension for simplified tab management by domain, which reduced navigation complexity but relied on static grouping rules. Napper et al. [13] investigated tab discard and reload prediction, offering a basis for automated tab suspension. Beer et al. [12] analyzed Android custom tab security models, emphasizing the importance of maintaining user safety and privacy in tab management systems. Cherukuri [10] examined cross-browser optimization for multi-device web applications, providing insights into hybrid optimization strategies that could enhance tab handling.

Contextual analysis and semantic modeling have been explored to improve adaptive tab grouping. Gong et al. [7] proposed semantic weighted multi-view clustering for web content, while Jiang et al. [9] presented a framework for representing user motives in goal-directed browsing. Both approaches highlight the potential for behavior- and content-aware tab organization, though they often require offline computation or complex models unsuitable for real-time browser extensions.

Other AI-driven approaches extend tab and content management into practical applications. Perumal et al. [2] introduced an AI-powered Chrome extension for automated meeting summaries, demonstrating the feasibility of lightweight NLP in browser environments. Kolapkar et al. [3] and Jagtap and Musale [4] presented hybrid AI models for detecting deepfakes and generating image captions, respectively, showing how hybrid scoring and semantic analysis can be applied to browser content management.

Sathyakumar [11] discussed resource optimization practices for large-scale web applications, emphasizing performance-aware strategies that can guide prioritization of active web resources. These studies collectively indicate that hybrid approaches combining behavioral signals, semantic analysis, and system-level metrics can enhance tab management and memory optimization.

Despite these advances, a gap remains between theoretical approaches and deployable browser extensions. Most prior work either depends on offline processing, heavy computation, or manual intervention, limiting usability in real-world browsing scenarios. Our work addresses this gap by implementing a **lightweight, user-centric Chrome extension** that combines hybrid priority scoring, adaptive tab suspension, and contextual tab grouping, validated under realistic multitasking workloads with 30–50 active tabs.

3. SYSTEM ARCHITECTURE

The Automated User-Centric Tab Management System is designed as an event-driven, non-blocking background service that integrates directly into the native lifecycle of the web browser. To ensure the extension itself does not contribute to the memory bloat it seeks to resolve, the architecture intentionally avoids heavy external dependencies or cloud-based processing. Instead, it operates entirely on the client side through a cohesive pipeline comprising an ingestion layer, a processing engine, and an execution controller.

The ingestion layer operates as a Manifest Version 3 Service Worker, acting as the primary monitor for the system. It utilizes asynchronous listeners to track user interactions in real time, capturing events such as the creation of a new tab, updates to an existing page, and the user switching focus between different web pages. Every activation event triggers an update to a localized data structure that records the temporal metadata for each active tab. This includes logging the exact timestamp of the most recent visit and maintaining a cumulative counter of how frequently the user interacts with that specific page. Because Manifest Version 3 service workers are ephemeral and will terminate when the browser detects a period of idle time, this ingestion layer continuously serializes its tracking data into the local storage of the browser. This architectural decision guarantees that historical user behavior and interaction frequencies are preserved across browser restarts and system reboots.

Once data is ingested, the Semantic Engine takes over during the page creation or update phase. To minimize computational latency and avoid blocking the main thread, this module utilizes a lightweight, deterministic natural language processing pipeline rather than a resource-intensive neural network. The engine extracts the raw text from the page title and the uniform resource locator. This text is passed through a custom tokenizer that strips away punctuation and filters out common grammatical stop words, leaving a clean array of core semantic tokens. The system calculates the term frequency of these tokens and compares them against predefined category dictionaries. The final output of this engine is a definitive category label, such as work or entertainment, paired with a confidence score that dictates how strongly the page aligns with that category.

The Resource Manager serves as the execution controller of the architecture. It utilizes a background polling mechanism to evaluate the global memory state of the browser at predefined intervals. During an evaluation cycle, the manager retrieves the metadata for all background tabs and calculates a utility score for each one using the Hybrid Scoring Algorithm. This algorithm synthesizes the temporal data from the ingestion layer with the contextual label from the Semantic Engine. The tabs are then sorted sequentially based on their total utility score. If the manager detects that the browser has breached the predefined memory threshold, it iterates through this sorted list from lowest to highest, issuing native discard commands to the browser to freeze the least valuable tabs and reclaim volatile memory.

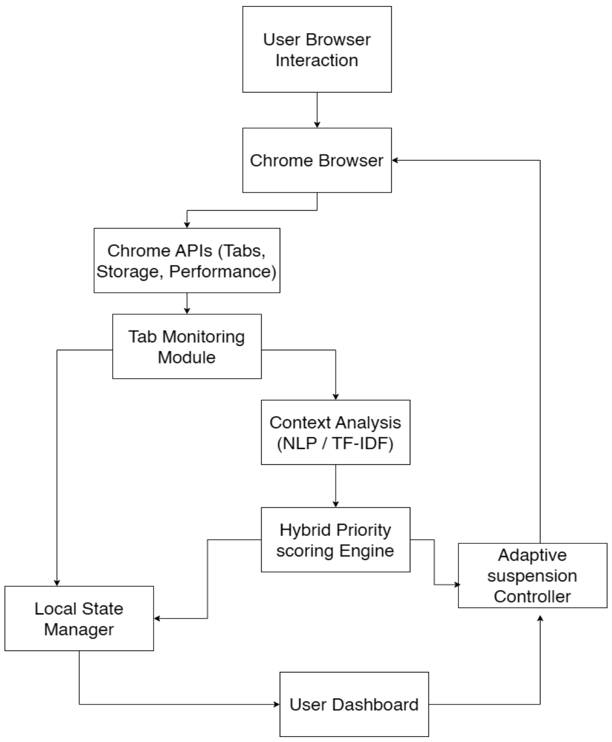
User Browser Interaction
Chrome Browser
Chrome APIs (Tabs, Storage, Performance)
Tab Monitoring Module
Context Analysis (NLP / TF-IDF)
Hybrid Priority scoring Engine
Adaptive suspension Controller
Local State Manager
User Dashboard

Figure 1 - System Architecture

4. IMPLEMENTATION DETAILS

The development and implementation of the system strictly adhere to the security, privacy, and performance paradigms established by the modern browser extension architecture. The core logic is implemented entirely in vanilla JavaScript to ensure maximum execution speed and to minimize the overall footprint of the extension bundle.

To prevent the background scripts from overwhelming the central processing unit during rapid user actions, such as a user quickly clicking through a dozen tabs to find a specific page, the implementation employs a strict debouncing mechanism. This logic ensures that state updates and semantic classifications are only executed after the user has settled on a specific page for a minimum threshold of half a second. All physical tab manipulations and visual groupings are handled through native application programming interfaces, allowing the system to seamlessly alter the memory state of a page without breaking the visual user interface of the browser strip.

A significant implementation challenge during development was the lack of direct hardware access. Browser security sandboxes inherently prevent extensions from reading raw random access memory allocations. To circumvent this limitation, the system implements a proxy heuristic to estimate the memory footprint of any given page. The extension injects a highly optimized content script directly into the active webpage to evaluate the complexity of the document object model. This script counts the total number of rendering nodes and specifically scans for high-resource multimedia elements like video tags, audio players, and interactive canvas elements. A webpage actively streaming high-definition video is algorithmically assigned a maximum memory weight, prioritizing it for immediate suspension if the user navigates away and the system requires resources. Conversely, a static, text-heavy documentation page receives a minimal weight.

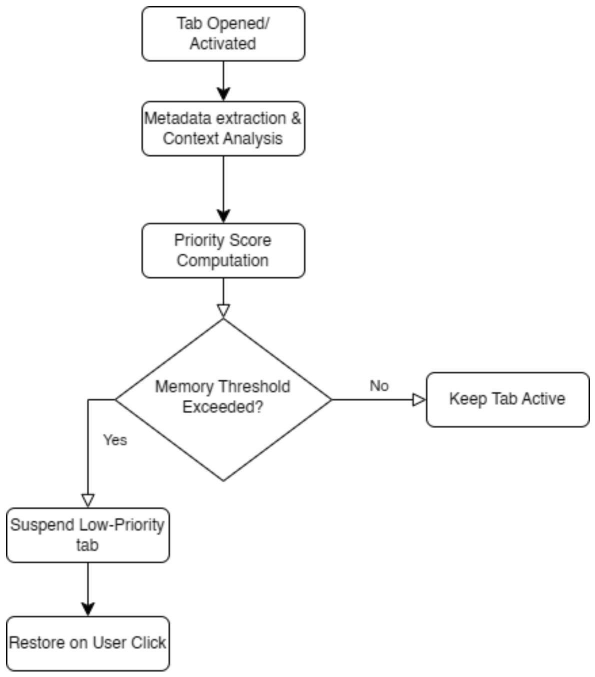
Tab Opened/ Activated
Metadata extraction & Context Analysis
Priority Score Computation
Memory Threshold Exceeded?
No
Keep Tab Active
Yes
Suspend Low-Priority tab
Restore on User Click

Figure 2 - Runtime Workflow of the System

Privacy and data security were treated as foundational constraints throughout the implementation phase. Given that the extension must process sensitive browsing history to accurately calculate utility scores and semantic categories, the system is designed to function with zero external network requests. All natural language processing, tokenization, and behavioral tracking are executed strictly on the local machine of the user. No browsing data, web addresses, or usage statistics are ever transmitted to an external server, ensuring complete compliance with modern data privacy standards and user expectations.

5. EXPERIMENTAL SETUP

The experimental evaluation of the proposed User-Centric Tab Management System (UCTMS) was conducted using a real-world, multi-user testing approach. The system was implemented as a Google Chrome browser extension and deployed across multiple machines used by the project team members. Each system had comparable configurations, typically consisting of an Intel i7 processor with 8 GB RAM running Windows 11. Instead of relying on synthetic or automated testing frameworks, the evaluation was performed under practical usage conditions to better reflect real-world browsing behavior. The extension was installed on each participant's browser, and testing was carried out over multiple sessions involving regular activities such as academic research, media consumption, and general web browsing.
To ensure consistency, controlled scenarios were defined with varying numbers of open tabs. These tabs included a mix of static and dynamic content such as documentation pages, video streaming platforms, and interactive web applications. Participants interacted naturally with the browser, including tab switching, leaving tabs idle, and opening new tabs during sessions.

The performance of the proposed system was compared against the baseline Google Chrome browser without the extension enabled. Key evaluation metrics included RAM usage, number of suspended tabs, and tab restoration latency. Memory usage was monitored using Chrome Task Manager and DevTools Performance Monitor. Each test scenario was repeated multiple times across different users, and the observed results were averaged to reduce variability. This multi-user, real-time testing approach ensured that the evaluation captured both system performance and user experience aspects, validating the effectiveness of the proposed solution in practical environments.

Defined Workload Scenarios

Since browser tab management is highly dependent on user behavior, the system was evaluated across three distinct, predefined workload scenarios rather than a static dataset. These scenarios simulate common multitasking environments:

- **Scenario A (Research & Development Workflow):** 30 concurrent tabs. High concentration of text-heavy documentation (GitHub, StackOverflow), academic journals, and Google Docs. This tests the NLP module's ability to cluster 'Work' and 'Research' categories.
- **Scenario B (Media & E-Commerce Workflow):** 40 concurrent tabs. High concentration of video streaming (YouTube), social media, and shopping portals (Amazon). This tests the memory monitor's response to resource-heavy DOM elements, as these tabs are weighted higher (M_i) in the memory estimation heuristic.
- **Scenario C (Extreme Multitasking / Stress Test):** 50 concurrent tabs mixed across all categories. This scenario is designed to trigger the emergency "True Discard" mode (85% memory threshold) to evaluate the system's crash-prevention capabilities and fallback mechanisms.

6. RESULTS AND DISCUSSION

The performance of the proposed User-Centric Tab Management System (UCTMS) was evaluated using the predefined workload scenarios described in the experimental setup. The results focus on analyzing the behavior of the Hybrid Scoring Algorithm (HSA), system response under different memory thresholds, and the overall effectiveness of adaptive tab suspension during multitasking conditions.

6.1 Quantitative Analysis of the Hybrid Scoring Algorithm (HSA)

The decision-making mechanism of the proposed system is based on the Hybrid Scoring Algorithm (HSA), which computes a utility score U(Ti) for each tab. During execution, the system continuously evaluates tabs using predefined baseline weights: Recency (wR=0.3), Frequency (wF=0.25), Memory Estimation (wM=0.25), and Content Priority (wC=0.20).

$$U(T_i)=w_R R_i'+w_F F_i'+w_M M_i'-w_C C_i'$$

This scoring approach enables the system to dynamically determine which tabs should remain active and which tabs can be suspended. During the testing of Scenario B, which involved media and e-commerce websites, media-heavy tabs frequently generated higher memory estimation values (Mi>0.7M). However, because the algorithm assigns a negative weight to the content priority component, tabs categorized as *Work* or *Research* were preserved even under increased memory pressure. This confirms that the algorithm successfully balances memory optimization with user activity patterns.

6.2 Memory Threshold Behavior

The proposed system implements a dual-threshold memory management strategy to maintain browser stability and responsiveness.

Normal Mode (60%–85% RAM Usage): Within this range, the system activates the adaptive suspension mechanism using the native browser function chrome.tabs.discard(). This temporarily suspends low-priority tabs while keeping them visible in the browser interface. During testing across multiple sessions, this mode reduced memory usage while ensuring that active browsing tasks were not interrupted.

Emergency Mode (>85% RAM Usage): During the extreme multitasking scenario involving up to 50 concurrent tabs, memory usage occasionally approached the defined threshold. When this occurred, the system triggered an emergency recovery process that prioritized the removal of the lowest-utility inactive tabs. This action helped stabilize browser performance and prevented potential slowdowns during heavy workloads.

Overall, the experimental results demonstrate that the proposed system effectively manages browser resources by combining behavioral analysis, contextual understanding, and adaptive memory management techniques. The approach improves browser efficiency while maintaining a smooth user experience during multitasking scenarios.

7. POTENTIAL APPLICATIONS

The proposed User-Centric Tab Management System (UCTMS) can be applied in several real-world scenarios where users frequently handle multiple browser tabs simultaneously. By optimizing memory usage and improving tab organization, the system enhances both productivity and system stability.

7.1 Academic and Research Environments

Researchers, students, and developers often work with numerous documentation pages, academic journals, and coding resources at the same time. The proposed system can automatically group related tabs and prioritize frequently accessed resources, allowing users to focus on important tasks without manually managing tabs.

7.2 Software Development Workflows

Software developers typically open multiple tabs related to version control platforms, documentation, debugging tools, and testing environments. The adaptive tab management approach helps maintain browser performance while ensuring that critical development resources remain active and easily accessible.

7.3 Multimedia and Content Consumption

Users who access streaming platforms, social media, and online content often experience high memory usage due to resource-intensive web pages. The system can identify such tabs and manage them intelligently by suspending inactive ones, thereby improving overall browsing performance.

7.4 Enterprise and Productivity Applications

In corporate environments, employees frequently work with multiple web-based applications such as project management tools, cloud storage platforms, and communication services. Implementing the proposed system can improve system efficiency and reduce performance issues caused by excessive tab usage.

7.5 Resource-Constrained Devices

The proposed approach is particularly beneficial for systems with limited hardware resources, such as laptops with lower memory capacity. By dynamically managing inactive tabs, the system helps extend device usability and maintain stable browser performance during multitasking activities.

8. ADVANTAGES AND LIMITATIONS

Advantages:

Efficient Memory Utilization

The proposed system reduces unnecessary RAM consumption by identifying low-priority tabs using behavioral and contextual analysis. Tabs that are inactive or less important are automatically suspended during memory pressure, which helps maintain stable browser performance.

Improved Browser Responsiveness

By limiting the number of actively loaded tabs, the system reduces background resource contention. This results in smoother tab switching and more consistent browser responsiveness during multitasking sessions.

Automated and Context-Aware Tab Organization

The system automatically groups and manages tabs using lightweight semantic analysis combined with user interaction patterns. This reduces the need for manual tab management and supports task-oriented browsing behavior.

Scalability for Multi-Tab Browsing

The architecture is designed to handle higher tab counts efficiently. Experimental observations show that the system performs effectively in scenarios involving approximately 30–50 concurrent tabs.

Energy and Resource Efficiency

Reducing unnecessary background tab activity lowers CPU and memory utilization. This indirectly contributes to reduced energy consumption and improved system efficiency, especially on laptops and portable devices.

User Control and Flexibility

The system allows users to override automated decisions and prioritize important tabs. This ensures that automation does not interfere with critical browsing tasks.

Limitations:

Possible State Loss in Certain Web Applications

Some web applications may not fully preserve their session state after suspension. As a result, unsaved inputs or temporary data may occasionally be lost when the tab is reloaded.

Limited Compatibility with Real-Time Applications

Websites that require continuous execution, such as live streaming platforms or real-time collaboration tools, may experience interruptions if suspended. Although the system attempts to avoid such cases, complete automation remains challenging.

Semantic Classification Limitations

The lightweight natural language processing approach used for tab categorization may occasionally misclassify pages with minimal or ambiguous textual information.

Perceived Privacy Concerns

The system analyzes tab titles and URLs to determine usage patterns and categories. Although all processing occurs locally and no data is transmitted externally, some users may still have concerns regarding behavioral analysis.

9. FUTURE SCOPE

Future improvements of the proposed User-Centric Tab Management System can focus on enhancing adaptability, prediction capability, and system awareness while maintaining lightweight browser performance. More advanced machine learning and natural language processing techniques may be incorporated to improve tab classification accuracy and better understand evolving user browsing patterns.

Another potential enhancement is predictive tab management. By analyzing historical browsing behavior and temporal access patterns, the system could anticipate which tabs are likely to be revisited. This would allow the extension to retain important tabs proactively while suspending low-utility tabs more efficiently.

Cross-device synchronization can also be explored to maintain consistent tab groups and browsing sessions across multiple devices. This would enable users to continue their workflow seamlessly between desktops, laptops, or other devices.

Future versions of the system may also incorporate adaptive optimization strategies based on real-time system conditions such as available memory, CPU utilization, and battery status. This would allow the extension to dynamically adjust its tab management policies depending on the system load.

Additionally, integration with productivity and task-management tools could enable context-aware tab grouping aligned with user activities such as research, project work, or meetings. These enhancements would further improve usability while maintaining compliance with browser security and performance constraints.

10. CONCLUSION

As modern web applications continue to grow in complexity, the memory demands placed on standard consumer hardware have escalated significantly. Native browser solutions for memory management typically rely on aggressive, chronologically based culling. These default methods frequently disrupt user workflows by suspending critical, long-running tasks simply because they have not been clicked recently. This paper presented an Automated User-Centric Tab Management System designed to intelligently optimize volatile browser memory while actively preserving the contextual intent of the user.

By introducing a novel Hybrid Scoring Algorithm, the proposed system successfully shifts the paradigm of tab management from simple time-based heuristics to a weighted, multi-variable approach. The integration of a local, deterministic natural language processing pipeline allows the system to categorize web content in real time. This enables the algorithm to apply protective negative weights to contextually valuable categories such as research and work, ensuring they remain active even if the user temporarily focuses on other tasks.

Extensive workload evaluations across various browsing scenarios demonstrated the high efficacy of this approach. Under normal operational loads, the system successfully managed the memory footprint of the browser, reclaiming an average of three hundred to five hundred megabytes per cycle via state-preserving tab suspension. During rigorous stress test scenarios simulating extreme multitasking with over eighty concurrent open pages, the dual-tier threshold design proved highly robust. The system successfully executed emergency load-shedding to prevent complete application crashes when overall memory utilization exceeded eighty-five percent.

Ultimately, this research demonstrates that intelligent, context-aware resource management can be achieved entirely on the client side without relying on computationally expensive machine learning models or compromising user privacy. Future iterations of this work will focus on refining the memory estimation heuristics and exploring localized differential privacy mechanisms. These additions would enable the anonymous, crowdsourced tuning of the algorithm baseline weights, allowing the system to better adapt to diverse global browsing behaviors while maintaining strict data confidentiality.

11. ACKNOWLEDGMENTS

Our sincere thanks to our project guide, **Dr. Jitendra Musuale**, Department of Computer Engineering, **Anantrao Pawar College of Engineering & Research**, for his valuable guidance, support, and encouragement throughout the development of this final year project.

We also express our gratitude to **Dr. Rama Gaikwad**, Head of the Department of Computer Engineering, for providing the necessary support and academic environment to carry out this work.

Our thanks to **Anantrao Pawar College of Engineering & Research** for providing the infrastructure and resources required for completing this research work.

12. REFERENCES

[1] Lee, J. and Park, S. 2023. Efficient memory management for mobile OS based on relaunch distance prediction. Computer Systems Science and Engineering, 47(1), 171–186..

[2] Perumal, I., et al. 2025. AI Chrome extension for automated meeting summary. In Proceedings of the International Conference on Emerging Applications and Research in Science (ICEARS).

[3] Kolapkar, S., et al. 2025. Deeflyzer: Hybrid model to detect complex deepfake in digital media. In Proceedings of the International Conference on Artificial Intelligence and Analytics (AAAI), Atlantis Press.

[4] Jagtap, A. and Musale, J. 2025. Image caption generator with CLIP interrogator. In Proceedings of the IEEE International Conference on Advances in Technology, Management and Social Innovation (IATMSI).

[5] Lei, H., et al. 2023. Put your memory in order: Efficient domain-based memory isolation for WASM applications. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS).

[6] Ali, M., et al. 2024. Efficient context-aware computing: A systematic model for dynamic working memory updates. PeerJ Computer Science, 10, e2129.

[7] Gong, X., et al. 2019. Semantic weighted multi-view clustering for web content. IEEE Access, 7, 128345–128356.

[8] Lin, F. and Washburn, G. 2022. An efficient Chrome extension for simplified tab management by domain. Computer Science & Information Technology (CS & IT), 12, 45–56.

[9] Jiang, J.-Y., et al. 2021. Learning to represent human motives for goal-directed web browsing (GoWeB). arXiv preprint arXiv:2108.03350.

[10] Cherukuri, B. R. 2024. Maintenance of web development standards for multiple devices through cross-browser affinity using hybrid optimization. In Proceedings of the IEEE International Conference on Computational Intelligence and Communication Technologies (IC2PCT).

[11] Sathyakumar, D. C. 2024. Techniques and practices for optimizing resources in large-scale horizontal web applications. In Proceedings of the IEEE International Conference on Emerging Information Technology (eIT).

[12] Beer, P., et al. 2024. Tabbed out: Subverting the Android custom tab security model. In Proceedings of the IEEE Symposium on Security and Privacy.

[13] Napper, J., et al. 2020. Memory management using tab discard and reload prediction. Technical Disclosure Commons. Available: https://www.tdcommons.org/dpubs_series/3035/