ESP IJAST
ESP International Journal of Advancements in Science & Technology
ISSN: 2583-9233/ Volume 4 Issue 4 October-December 2026
Paper Id: IJAST-V4I4P104/Page No: 25-32

Original Article

Designing Web Interfaces for Governed LLM Agents: A Review of MCP, Structured Access, and Human Control

Tanishka Dhanwan
Department of Computer Engineering, Arya College of Engineering & Information Technology, Kukas, Jaipur, Rajasthan, India.

Received Date: 18 October 2026 Revised Date: 24 October 2026 Accepted Date: 28 October 2026

Abstract: *Websites that can be used by language-model agents need more than a collection of API endpoints. They need compact observations, unambiguous actions, current state, authorization boundaries, accessible human fallbacks, and evidence that can be checked after execution. This review examines ten supplied studies covering agentic AI, Model Context Protocol (MCP), structured web access, human–AI dialogue, tool security, and domain-specific orchestration. It separates findings that recur across the literature from claims that remain dependent on a particular benchmark or implementation. The review then organizes the evidence into a layered design in which an LLM interprets and plans, while policy gateways and deterministic services validate, authorize, execute, and verify. It concludes with a research agenda for effectiveness, context cost, latency, security, accessibility, and maintainability. The paper does not claim that MCP alone improves every website.*

Keywords: *Model Context Protocol, Web Agents, Structured Interfaces, Agent Security, Accessibility, Human–AI Interaction*

INTRODUCTION

A web page is designed for a person who can interpret layout, labels, context, and visual state. An autonomous or semi-autonomous agent needs the same information in a different form: fields must be identifiable, actions must have typed inputs, state must have a lifetime, and failures must be distinguishable. The engineering problem is therefore not simply "how can a model click a button?" It is how a website can expose useful capabilities without handing an uncertain planner uncontrolled authority.

Model Context Protocol (MCP) is relevant because it gives hosts and clients a common way to discover and invoke server-side resources, prompts, and tools through a structured protocol 1. However, a protocol does not decide whether a particular action is safe, whether a user is entitled to perform it, or whether a returned value is current. Those decisions remain responsibilities of the application and its operating environment.

This paper revisits ten supplied studies and several standards-oriented sources with four questions. Which design patterns recur across the studies? Which reported improvements are tied to particular implementations? What should a web interface expose to an agent while preserving human control? How can a future experiment compare HTML, retrieval, MCP, and hybrid designs without confusing protocol effects with caching, prompts, schemas, or deterministic backends?

The contribution is a compact design framework. It combines a source-aware evidence assessment, a layered architecture, a request lifecycle with explicit safety gates, and a measurement plan. The manuscript deliberately distinguishes three kinds of statement: a finding reported by a cited study, a requirement derived from a standard or security guide, and a proposal that still requires empirical testing.

II. REVIEW METHOD AND EVIDENCE SYNTHESIS

This paper uses a structured narrative review. The corpus consists of ten project-supplied papers selected for direct relevance to agentic AI, MCP, structured web access, interaction quality, security, and domain-grounded orchestration. Each paper was examined for its objective, system or review method, task setting, reported measurements, contribution, and stated or identifiable limitations. Findings were then grouped into five review themes: delegated computation, compact context, human control, security boundaries, and evaluation validity. Official MCP documentation, WCAG 2.2, Core Web Vitals guidance, Schema.org, OWASP guidance, and BrowserGym were used to interpret the design implications of the reviewed studies.

This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/2.0/)

The review is qualitative rather than a meta-analysis. The papers use different datasets, models, baselines, domains, and outcome measures, so their numerical results cannot be pooled responsibly. Reported values are discussed as findings of the original studies, not as measurements produced by this review. The proposed architecture and research agenda are review-derived contributions and require future empirical validation.

A. What the supplied studies actually show

The studies do not form one homogeneous experiment. The agentic-AI landscape is broad and organizes application areas and risks, whereas the dialogue review examines interaction quality and user-centered outcomes 9 11. ARGUS, structural analysis, and the urban-resilience workflow are domain demonstrations in which language models coordinate tools or solvers 10 14 15. MCP-Web-Curl and webMCP propose mechanisms for obtaining or compressing web context 12 18. WebMall compares alternative agent interfaces in a controlled simulated commerce environment 16. The MCP security study examines server code and ecosystem practices rather than user task completion 17.

Despite these differences, a common pattern appears. A language model is most useful when it performs interpretation, decomposition, explanation, and repair. It is less suitable as the final authority for calculations, permissions, transaction state, or other decisions that can be checked by a domain service. ARGUS uses a learned ranking stage followed by power-flow verification; the structural-analysis system delegates numerical work to OpenSees; and the resilience workflow uses EnergyPlus and expert review. These cases support a general separation between proposing a computation and performing the authoritative computation.

The interface studies suggest a second pattern: reducing irrelevant context can improve agent work, but the mechanism is not always MCP. A smaller response, a better prompt, cached data, a specialized transaction function, and parallel requests can all reduce time or tokens. The WebMall results vary by task, and webMCP reports a context reduction strategy whose effect is entangled with its metadata format and client implementation 16 18. The defensible conclusion is conditional: structured access is promising when the task has stable entities, clear operations, and a reliable source of current state.

The security studies support a third pattern: model-visible metadata is part of the attack surface. Descriptions, error messages, retrieved documents, and page content can contain instructions that are irrelevant or hostile to the user's goal. A schema can constrain syntax, but it cannot establish that the requested action is authorized. A signature can establish who signed a manifest, but not that its permissions are appropriate. The gateway and backend must therefore enforce the rules independently of the model 17 3 4.

B. Evidence limits

The reported results should not be generalized beyond their test settings. Small case studies can demonstrate an architecture without establishing its reliability across domains. A solver-backed system may be more accurate because it calls the solver, not because of the protocol label. A token reduction may not produce a faster page if authentication, network setup, or client-side decoding becomes slower. A repository scan can find suspicious patterns without proving that an exploit works in production.

The present synthesis therefore does not pool percentages or rank MCP against every alternative. It uses the studies to derive hypotheses and controls for a future comparison. The most important hypothesis is not "MCP is faster." It is that typed, bounded capabilities can reduce observation and planning burden when their information is complete, current, and safe to invoke.

III. ARCHITECTURE FOR AN AGENT-READY WEBSITE

Figure 1 places the model inside, rather than above, a set of controls. The presentation layer serves people through semantic HTML and accessible interaction. A structured observation layer exposes identifiers, fields, provenance, and freshness. An MCP client and policy gateway mediate proposed actions. Server tools call domain services that own data and side effects. The application remains usable if the agent layer is unavailable.

The observation layer should expose stable identifiers rather than relying on display text or CSS position. It should state whether a value is observed, computed, estimated, or supplied by the user. A price needs currency and observation time; availability needs a validity rule; an account record needs an authorization context. Structured data vocabularies such as Schema.org can help describe entities and relationships, but they do not replace application authorization or transaction logic 7.

The capability layer should prefer narrow operations. `search_products`, `create_draft`, `preview_order`, and `check_status` communicate different risks more clearly than a generic `execute_request`. Each capability needs an input schema, output schema, version, allowed scope, side-effect classification, timeout, rate limit, idempotency rule, and error vocabulary. A read operation must be read-only because the server enforces it, not because the description says so.

The gateway is the trust boundary for proposed actions. It checks actor, tenant, target, scope, parameter limits, state freshness, and policy. For an irreversible operation, approval must bind to the exact target and arguments and expire after a defined interval. The backend repeats authorization checks because a client or model can be compromised. Audit records should capture the proposal, decision, execution result, and verification result with sensitive values redacted.

IV. REQUEST LIFECYCLE AND FAILURE SEMANTICS

A safe agent workflow is a state machine rather than a single model completion. Figure 2 describes the main path. Interpretation produces a draft plan. Validation checks both the schema and the meaning of the request. Policy decides whether the actor may perform the operation. Approval is requested only when the impact requires it. Execution occurs through an idempotent operation. Verification compares the result with the expected state before the system reports success.

The failure states are part of the interface contract. An ambiguous request should lead to clarification, not an inferred sensitive value. A stale quote should be revalidated, not presented as current. A timeout on a mutation should lead to a status lookup rather than an automatic duplicate. A partial result should identify what completed and what did not. A refusal should give a recovery path when one exists.

This lifecycle reflects the two-speed pattern discussed in the semantic-mediation study: deterministic checks can respond immediately, while generative planning is reserved for ambiguity and composition 13. The distinction is useful for web applications. Permission, scope, and input limits should not wait for a persuasive explanation. Conversely, a user may accept a slower planning phase when the system provides progress and a cancellation control.

V. REQUIREMENTS FROM STANDARDS AND THREAT MODELS

A. Security

The system should test more than direct prompt injection. A page can contain a sentence instructing the agent to disclose a database; a tool description can request secrets; an error message can suggest an administrator credential; a redirect can lead a fetch operation to an internal address. These inputs must remain untrusted data. The MCP security guidance identifies risks involving confused deputies, tokens, state, OAuth, and SSRF, while OWASP recommends least privilege, server-side authorization, output validation, logging, and recovery 3 4.

A practical test suite should include cross-tenant identifiers, wrong-audience tokens, expired approvals, replayed state handles, duplicate submissions, private network URLs, redirect chains, excessive downloads, and unbounded pagination. The evaluator should record whether the attack succeeded, whether legitimate work was blocked, whether information leaked, and whether recovery was possible. Security must be reported separately from task success; averaging an unsafe side effect with several successful read tasks would hide the critical failure.

B. Accessibility and human control

The agent layer must supplement the human interface. WCAG 2.2 provides the organizing principles of perceivability, operability, understandability, and robustness 5. In practice, a user should be able to navigate the workflow by keyboard, perceive focus, identify inputs and errors, review a confirmation, cancel an operation, and receive status updates in text. A visually efficient interaction is not sufficient if a screen-reader user cannot determine whether a job completed.

Evaluation should include keyboard-only operation, screen-reader announcements, focus behavior in dialogs, error association, reflow at high zoom, and recovery from dynamic updates. Human takeover should present the current state, completed actions, pending action, and relevant evidence. This avoids forcing a person to reconstruct the workflow from a long model transcript.

C. Performance and representation

Core Web Vitals provide familiar measurements for loading, interaction, and visual stability. The current guidance uses LCP, INP, and CLS thresholds at the 75th percentile, with good classifications of at most 2.5 seconds, 200 milliseconds, and 0.1 respectively 6. An agent experiment should also record tokens by component, number and type of tool calls, response bytes, browser steps, cache state, retries, and end-to-end latency.

"Efficient" must be decomposed. A smaller model prompt is context-efficient; it is not necessarily network-efficient. Fewer browser steps do not prove a faster backend. Lower latency does not prove correctness. Reporting a vector of measures is more informative than a single score. Device, browser, network, concurrency, cache, and model versions must accompany the measurements.

VI. REVIEW-DERIVED RESEARCH AGENDA

A controlled study should implement the same tasks over several interface conditions: semantic HTML, HTML plus structured metadata, RAG, MCP, and a hybrid design. Tasks should cover entity retrieval, vague search, multi-step forms, current availability, transaction preview, and recovery after a timeout or conflicting update. The fixtures should vary semantic quality, page size, dynamic behavior, localization, authentication, and adversarial content.

The primary outcome should be strict completion without an unsafe action. Secondary outcomes should include field correctness, freshness, context tokens, output tokens, tool calls, browser actions, bytes, p50 and p95 latency, retries, cost, accessibility failures, injection success, and repair. A trace should capture observations, model proposals, policy decisions, tool calls, backend responses, and final state. The final message alone is not evidence of success.

A factorial design can separate protocol from implementation effects. One factor can vary observation format; another can vary retrieval; another can vary deterministic verification; another can vary the policy gateway. The model and task remain fixed where possible. Results should report confidence intervals or uncertainty intervals appropriate to the unit of analysis. Workflow traces are not independent merely because they contain multiple tool calls.

The reproducibility archive should include fixture hashes, task definitions, prompts, model and provider versions, browser build, server revision, schemas, authorization configuration, network and cache settings, random seeds, redaction rules, scoring scripts, and an exclusion ledger. If live data cannot be redistributed, synthetic fixtures should be released and the difference described as a limitation. BrowserGym illustrates the value of repeatable web-agent environments, but domain tasks and safety criteria still need to be specified by the study authors 8.

VII. DISCUSSION AND LIMITATIONS

The synthesis points toward a governed form of agent readiness. The best interface is not the one with the most tools or the smallest response. It is the one that gives the agent enough structured information to act correctly, gives the server enough policy information to reject unsafe actions, and gives people enough evidence to understand and control the result.

MCP can be a useful interoperability layer in this design. It can make capabilities discoverable and typed, and it can connect an agent to services that perform authoritative work. It cannot guarantee least privilege, current data, safe metadata, accessible interaction, or correct business logic. Those properties belong to the surrounding application.

This review has limitations. It is a qualitative synthesis of heterogeneous papers and standards. It adds no human-subject study, live-site benchmark, penetration test, or new Core Web Vitals dataset. Several supplied studies use small scenarios or system-level comparisons, and their reported figures should remain attached to their original baselines. The framework should therefore be treated as a review-derived basis for a controlled study rather than as proof of universal superiority.

VIII. CONCLUSION

Agent-ready web design is best approached as a systems problem. Semantic HTML and structured data support observation; narrow typed capabilities support action; policy gateways protect authorization and side effects; deterministic services provide authoritative computation; verification and provenance support trust; accessible fallbacks preserve human agency. MCP can connect these parts, but it does not replace them.

As a review paper, this work's next step is a preregistered comparison of HTML, retrieval, MCP, and hybrid interfaces under matched tasks and explicit safety criteria. Such a study should report effectiveness, context cost, latency, accessibility, security, and maintainability separately. Until that work is completed, the strongest defensible conclusion is conditional: structured, governed capabilities may make some web-agent workflows more reliable and economical, provided that the protocol is supported by sound application design.

IX. IMPLEMENTATION DETAIL AND SOURCE-SPECIFIC IMPLICATIONS

A. From Evidence to Design Decisions

The source papers are most useful when their lessons are kept at the level where they were observed. The agentic-AI review is a reminder that adding tools changes the risk profile of a language model. Memory, planning, delegation, and external

actions create failure chains that do not exist in a one-turn answer 9. For a website, this means that an action log and a permission model should be designed before exposing a tool catalog. The system should be able to answer which capability was selected, why it was permitted, and what state changed.

The dialogue-quality review shifts attention from completion alone to the experience of correction and control 11. An agent that gives a correct answer but hides uncertainty can still produce poor interaction quality. A web interface should therefore expose the reason for a clarification, the evidence behind a recommendation, and the remaining choices. Explanations need not be long. A short statement such as “inventory was checked at 14:10 IST; the quote expires in five minutes” can be more useful than a generic confidence paragraph.

ARGUS contributes the idea of screening candidates before invoking an authoritative solver 10. In a website, a lightweight router can decide whether a request is explicit enough for a deterministic operation or requires language-model planning. The router should not be confused with a safety decision. It can reduce unnecessary planning, while the policy gateway still validates scope and side effects. The reported best-effort behavior under infeasible conditions is equally important: a system should say that no option satisfies all constraints instead of silently relaxing the user’s requirements.

The structural-analysis study demonstrates why typed orchestration matters in technical domains 14. A model can translate a request into a structured input, but a domain solver should calculate the result. The same distinction applies to a shopping total, account balance, eligibility result, or reservation. The user interface should label model-inferred fields, user-entered fields, and backend-authoritative values differently. This prevents a plausible sentence from being mistaken for a verified result.

The urban-resilience workflow shows the value of a reviewable artifact chain 15. Retrieval, generated configuration, tool execution, simulation output, and expert corrections should be preserved as separate stages. A website that uses an agent to configure a report, dashboard, or operational scenario should retain the draft, the approved version, and the final output. This is useful for audit and for debugging disagreements between the model and the domain expert.

The WebMall comparison is a warning against a single universal interface winner 16. Search for a vague or cheapest item may favor broad retrieval, while a transaction may favor a typed action with current validation. A benchmark should therefore use several task families and report per-task outcomes. Averages across unrelated tasks can conceal an important reversal, such as a method that is excellent for discovery but poor at commitment.

MCP-Web-Curl and webMCP both emphasize compact interaction with web resources 12 18. Their designs suggest that a client benefits from bounded responses, explicit truncation, stable action descriptions, and page-aware metadata. The maintenance cost must be included. A manifest that becomes stale after a deployment is worse than a slightly larger but truthful fallback. Every structured capability should therefore have an owner, version, freshness field, and invalidation path.

The semantic-mediation work supports a separation between rapid deterministic reaction and slower generative reasoning 13. This separation can be implemented in a gateway: simple reads and policy checks proceed through deterministic code, while ambiguous requests are routed to a planner. A model should not be allowed to skip the fast safety lane merely because it produced a confident explanation.

Finally, the MCP security study provides an operational reason to treat server descriptions and repository maintenance as part of the product 17. A registry should record who owns a server, what it can access, which revision is deployed, when it was scanned, and how it can be revoked. Tool metadata should be reviewed like code because it influences model behavior. A server that is easy to install but impossible to audit is not an efficient integration.

B. Capability Contract

The following fields are recommended for every agent-facing operation. The `name` and `version` identify the contract. `purpose` explains the user-visible outcome. `input` and `output` define types, required fields, units, bounds, and nullable values. `scope` identifies the minimum permission. `side_effect` states whether the operation reads, drafts, changes, communicates, or commits. `freshness` states how long returned values are valid. `errors` defines repairable and non-repairable states. `audit` identifies what is recorded. `recovery` identifies retry, cancellation, rollback, or takeover behavior.

The contract should also distinguish a resource from an action. A resource reports information; an action requests a change. A preview is not a commit. A queued job is not a completed result. These distinctions allow the interface to show an

accurate state to both the model and the person. They also make it possible to test that a supposedly read-only path cannot mutate data through an unexpected field.

Input validation should occur at three levels. Syntactic validation checks types and formats. Semantic validation checks whether values make sense in the domain. Authorization validation checks whether the actor may use those values on that resource. Returning one natural-language error for all three levels makes repair difficult. Field-level errors should identify the problem without exposing secrets or cross-tenant information.

Output validation is equally important. A tool should return a typed status such as `verified`, `pending`, `stale`, `partial`, `blocked`, or `failed`. It should include provenance and observation time where relevant. “Success” should be reserved for a verified outcome, not an acknowledgement that a request entered a queue. If the backend cannot verify the final state, the user should see that limitation.

C. Concrete Safety Scenarios

A product page contains an instruction telling the model to send customer records to an external email address. The correct behavior is to treat the instruction as page content. The model may summarize the page, but the unrelated email capability should not be available for the user’s shopping task. This test checks the separation between evidence and instructions.

A user approves a reservation for one hotel and the model later changes the hotel identifier while keeping the same approval. The gateway should calculate a parameter hash and reject the changed request. This is an approval-binding test. A broad statement such as “book the hotel” should not authorize every property or price.

A fetch tool receives a URL that resolves to a private network address. The server should block the request, limit redirects, and log the policy decision. The model should receive a typed network-policy error rather than an opportunity to try alternative encodings. This test is relevant even when the tool is described as read-only.

A long-running simulation times out after the server may have accepted the job. The client should query the operation identifier and report pending, completed, or failed state. It should not submit a second job automatically. This scenario tests idempotency, progress, and recovery together.

A form is usable by mouse but its confirmation dialog cannot be reached by keyboard. The agent may complete the task, but the website fails the human-control requirement. Accessibility is not an optional presentation enhancement; it is part of the system’s ability to let a person inspect or take over an agent workflow 5.

D. Reporting Protocol

Every evaluation run should begin from a documented site state. The record should identify the fixture, user role, locale, time zone, browser, viewport, cache state, network profile, model version, prompt version, schema revision, and server revision. Each observation and action should receive a timestamp and an event type. Sensitive values should be replaced by consistent redaction tokens before storage.

A run ends when the goal is verified, the agent stops, the timeout is reached, a human takes over, or a safety rule interrupts it. These outcomes should not be collapsed into one failure category. Infrastructure failure, model misunderstanding, authorization rejection, stale data, unsafe action, and inaccessible control require different fixes. The analysis should publish the exclusion ledger so that readers can see which runs were removed and why.

A strict success metric is appropriate for a safety-critical action: all required conditions must hold and no prohibited side effect may occur. Partial credit is useful for diagnosis but should not hide an unsafe commit. Results should include uncertainty estimates and the number of independent workflows. If human raters judge final states or accessibility, a subset should be scored by two raters and disagreements resolved using a predefined rule.

E. Originality and Citation Practice

A synthesis is original when it contributes its own organization, comparison, interpretation, and design implications while crediting the evidence used. Technical names and study titles must remain recognizable so that readers can locate the sources. Surrounding explanatory prose should be written from the authors’ own analysis rather than copied sentence patterns. Numerical values should be attributed to the study that reported them, with its baseline and limitations.

The present review does not claim that a plagiarism checker will produce a particular percentage. Automated similarity tools can flag titles, bibliographies, standards terminology, and common technical language. They can also miss paraphrased

copying. The appropriate academic response is source-aware writing, accurate citation, and manual review of flagged passages. The authors should still submit the final PDF to the college's approved checker and inspect the detailed report if access becomes available.

X. DEPLOYMENT TRADE-OFFS AND INTERPRETATION

Not every website needs an MCP server. A static information site with clear headings, accessible markup, and useful structured data may gain little from adding a transactional tool layer. A retrieval index may be the appropriate addition when the problem is finding relevant passages across a large corpus. MCP becomes more attractive when the site owns authoritative operations, needs typed parameters, or must expose a reusable capability to several hosts. This decision should be made per task rather than per brand or framework.

A useful maturity path begins with semantic HTML and stable identifiers. The next stage adds structured read-only resources with provenance and freshness. Later stages add draft and preview operations, followed by governed commits with approval, idempotency, audit, and verification. Long-running jobs can then add progress, cancellation, and human takeover. Advancing gradually makes it possible to observe whether a new layer solves a real bottleneck before accepting its maintenance and security cost.

The hybrid design is usually the most practical. Humans receive a normal accessible website. Agents may use structured resources for discovery, retrieval for broad search, and MCP tools for current or state-changing operations. The application chooses the authoritative source for each value. A search index can locate a product, but an inventory service must verify stock. A model can draft a document, but the content system must decide whether the user may publish it. A page can describe a service, but the server must enforce its limits.

Interpretation of benchmark results should follow the same discipline. If MCP uses fewer tokens, inspect whether it was given a smaller observation, a better prompt, a cached resource, or a custom function. If MCP finishes faster, identify whether the gain came from fewer browser actions, parallel calls, a faster server, or reduced model reasoning. If it is safer, identify the control that caused the reduction: least privilege, approval, validation, or a restricted fixture. This decomposition turns a product comparison into an explanation of mechanisms.

The study should also report negative results. A structured interface may increase page complexity, produce stale manifests, or perform worse during login and recovery. A retrieval system may find broad candidates but fail to establish current availability. An MCP tool may be efficient for a known action but brittle when the user is vague. Publishing these cases is important because agent-ready design is a trade-off among coverage, speed, maintainability, and control.

For a college-scale prototype, a modest fixture is sufficient if it is explicit. The authors can implement a small catalog, a form workflow, a read-only retrieval path, a preview action, and a deliberately malicious page. The useful contribution would be a traceable comparison with controlled conditions, not an inflated claim based on a large but opaque demo. The same fixture can support accessibility checks, security cases, latency measurements, and manual review of generated explanations.

The final manuscript should state the status of this work accurately. It is a synthesis and framework based on supplied literature and standards. It does not report a newly conducted benchmark or human-subject experiment. This limitation is not a weakness to hide; it defines the next research step and prevents the cited studies from being presented as evidence for claims they did not test.

Figure 1. Layered architecture: semantic web interface, structured observations, MCP client and policy gateway, domain tools, and deterministic services.

Figure 2. Validated request lifecycle: interpretation, validation, authorization, approval, execution, verification, recovery, and human takeover.

XI. REFERENCES

[1] Sarfraz Brohi and Qurat-ul-ain Mastoi and N. Z. Jhanjhi and Thulasyammal Ramiah Pillai. "A Research Landscape of Agentic AI and Large Language Models: Applications, Challenges and Future Directions." Algorithms, 2025.

[2] Suhani Mohanty and Rayner Dcunha and Marek Z. Reformat and Petr Musilek. "ARGUS: An MCP-Based Agentic LLM Architecture for Distribution Grid Decision Support." 2026.

[3] Luca Marconi and Luca Longo and Federico Cabitza. "Assessing Interaction Quality in Human--AI Dialogue: An Integrative Review and Multi-Layer Framework for Conversational Agents." Machine Learning and Knowledge Extraction, 2026.

[4] Rayhan Zahwan Saleh and Muharman Lubis. “Design and Implementation of MCP-Web-Curl: A Model Context Protocol Server for Web and API Access in Agentic Coding Assistants.” Jurnal Teknik Informatika, 2026.

[5] Andrei Kojukhov and Ilya Levin and Arkady Bovshover. “From Subsumption to Semantic Mediation: A Generative Orchestration Architecture for Autonomous Systems.” Algorithms, 2025.

[6] Carlos Avila and Daniel Ilbay and David Rivera. “Human--AI Teaming in Structural Analysis: A Model Context Protocol Approach for Explainable and Accurate Generative AI.” 2025.

[7] Jingfeng Zhou and Jiying Chen and Meng Wang and Baihua Dong and Xiaochi Zhou. “Large Language Model Orchestrated Workflow Integrating Retrieval-Augmented Generation and Model Context Protocol Tools for Urban-Scale Simulation-Based Resilience Assessment.” Building and Environment, 2026.

[8] Aaron Steiner and Ralph Peeters and Christian Bizer. “MCP vs RAG vs NLWeb vs HTML: A Comparison of the Effectiveness and Efficiency of Different Agent Interfaces to the Web.” 2026. DOI: 10.1145/3774904.3792893.

[9] Mohammed Mehedi Hasan and Hao Li and Emad Fallahzadeh and Gopi Krishnan Rajbahadur and Bram Adams and Ahmed E. Hassan. “Model Context Protocol (MCP) at First Glance: Studying the Security and Maintainability of MCP Servers.” 2026.

[10] D. Perera. “WEBMCP: Efficient AI-Native Client-Side Interaction for Agent-Ready Web Design.” 2026.

[11] Model Context Protocol. “Model Context Protocol Specification.” 2025. Available: https://modelcontextprotocol.io/specification/2025-06-18.

[12] Model Context Protocol. “Architecture Overview.” 2026. Available: https://modelcontextprotocol.io/docs/2026-07-28/learn/architecture.

[13] [13] {Model Context Protocol. “Security Best Practices.” 2026. Available: https://modelcontextprotocol.io/docs/2026-07-28/tutorials/security/security_best_practices.

[14] {OWASP Foundation. “AI Agent Security Cheat Sheet.” 2026. Available: https://cheatsheetseries.owasp.org/cheatsheets/AI_Agent_Security_Cheat_Sheet.html.

[15] {World Wide Web Consortium. “Web Content Accessibility Guidelines (WCAG) 2.2.” 2024. Available: https://www.w3.org/TR/WCAG22/.

[16] {Google Chrome Developers. “How the Core Web Vitals Metrics Thresholds Were Defined.” 2025. Available: https://web.dev/articles/defining-core-web-vitals-thresholds.

[17] {Schema.org Community. “Schema.org.” 2026. Available: https://schema.org/.

[18] Thibault Le Sellier De Chezelles and Maxime Gasse and Alexandre Drouin and others. “The BrowserGym Ecosystem for Web Agent Research.” arXiv preprint arXiv:2412.05467, 2024. Available: https://arxiv.org/abs/2412.05467.