# Principles of Designing Scalable Frontend Architectures for Integration with Artificial Intelligence Systems

[1] Goel Taran
[1] Expert in Frontend Engineering, USA

Received: 18th Nov 2025 | Received Revised Version: 28th Nov 2025 | Accepted: 17th Dec 2025 | Published: 26th Dec 2025

Volume 07 Issue 12 2025 | Crossref DOI: 10.37547/tajet/Volume07Issue12-14

## Abstract

*The article is devoted to the analysis and systematization of principles for designing scalable frontend architectures aimed at effective integration with artificial intelligence (AI) systems. The relevance of the study is determined by the exponential growth of the use of AI technologies, including generative models, in user interfaces, which generates new, increased requirements for the flexibility, performance, and fault tolerance of front-end systems. The scientific novelty of the work consists in the formulation of a comprehensive architectural model based on the author's practical experience in the domain of AdTech/MediaTech platforms. Within the framework of the study, the main challenges of integrating AI into the frontend are identified and structured, including state management, rendering of dynamic content, and ensuring low response latency. Contemporary design approaches are analyzed, including micro-frontends, server-side rendering, and API-first design. Particular emphasis is placed on the principles of system decomposition, performance optimization, and compliance with digital accessibility requirements. The purpose of the work is to develop and theoretically substantiate a set of architectural principles intended for building scalable frontend systems capable of natively interacting with AI services. To achieve this goal, methods of systems analysis of scientific literature, comparative analysis of architectural patterns, as well as the case study method based on the author's practical experience, are employed. In conclusion, the proposed modular AI-integrated architecture (MAI-FA) is presented, and conclusions are formulated regarding its applicability in the context of high-load and complex web systems. The findings presented in the article will be of interest to frontend architects, lead developers, and technical managers involved in the design of complex web applications with intensive use of AI.*

Keywords: frontend architecture, artificial intelligence, scalability, systems design, AI integration, micro-frontends, AdTech, generative AI, web application performance, architectural patterns, digital accessibility (a11y).

**Cite This Article:** Goel Taran. (2025). Principles of Designing Scalable Frontend Architectures for Integration with Artificial Intelligence Systems. The American Journal of Engineering and Technology, 7(12), 132–139. https://doi.org/10.37547/tajet/Volume07Issue12-14.

## 1. Introduction

Over the past decade, the field of web development has experienced a qualitative, essentially tectonic, shift driven by the ubiquitous integration of artificial intelligence systems. AI has ceased to function exclusively as a backend component and is increasingly moving into the user interface layer, where it provides personalized experiences, generates real-time content (GenAI), and orchestrates complex user interaction scenarios [3]. At the same time, traditional monolithic frontend architectures, designed for a relatively predictable and static user interface, demonstrate limited suitability to the new conditions. Embedding AI services requires the frontend not only to perform the basic function of data visualization, but also to exhibit properties of elasticity, fault tolerance, and high performance, as well as the ability to handle asynchronous data streams, dynamically generated interface components, and low-latency modes of operation [2, 4]. These limitations are most acute in high-load domains such as advertising (AdTech) and media platforms, where rendering speed and the ability of the system to adapt to content variations directly correlate with financial performance indicators.

The aim of this study is to propose and theoretically substantiate an integrated set of principles for designing scalable frontend architectures oriented toward integration with artificial intelligence systems. To achieve this aim, the following objectives must be addressed: to analyze contemporary scientific literature in order to identify the key problems and challenges arising from embedding AI services into frontend applications; to systematize existing architectural approaches (micro-frontends, SSR, component-based architectures) and performance optimization strategies (code splitting, lazy loading) from the perspective of their relevance to AI integration tasks; and, based on the analysis of literary sources, to formulate an original model (framework) and a set of key principles for designing a scalable, AI-oriented frontend architecture.

The scientific novelty of the study consists in the formulation of a holistic architectural model (MAI-FA) which, in contrast to prevailing approaches focused on individual aspects (for example, solely on performance or only at the API level), defines a comprehensive solution that combines the principles of modularity, independent deployment, dynamic user interface generation, and proactive performance management.

The working hypothesis of the author is that the use of a decomposed, service-oriented architecture (primarily micro-frontends), based on clearly specified API contracts and supplemented by modern techniques for build and rendering optimization (Tree Shaking, SSR), represents the most effective strategy for constructing scalable frontend systems capable of natively and with high performance integrating with AI services that generate the user interface.

## 2. Materials and Methods

The theoretical and methodological basis of the study is the systems approach, which considers frontend architecture as a complex sociotechnical system in which technological, organizational, and product-related aspects are interrelated. The core body of sources was formed from scholarly literature published mainly in recent years, when a qualitative growth has been observed in studies devoted to the integration of AI, generative models, and high-load web interfaces. The materials analyzed included publications on scalable frontend architectures, architectural patterns (micro-frontends, SSR, hybrid rendering), systems design of AI applications, as well as industry practices of AdTech/MediaTech platforms. A substantial part of the empirical base is the author's practical experience in designing high-load frontend systems in the AdTech industry, interpreted within a formalized case methodology.

The selection of literature was carried out according to the principles of relevance, novelty, and methodological soundness. The review included works that simultaneously satisfied at least two conditions: they analyze architectural solutions for the frontend or web applications; they address issues of integrating AI/ML services, generating or personalizing the user interface, or optimizing performance under conditions of high content dynamics. Additionally, publications were taken into account that describe practices of implementing micro-frontends, design systems, an API-first approach, and build optimization strategies, since these approaches directly correlate with the task of creating AI-oriented architectures. Sources that focus exclusively on backend or infrastructure aspects (orchestration, storage, purely model-centric ML tasks) without an explicit connection to the frontend, as well as works that do not contain reproducible methodological foundations, were consistently excluded from consideration.

## 3. Results

This section presents the results of an analysis of the practical application of architectural principles based on the development of high-load frontend systems in the AdTech and MediaTech domains. Professional work in the companies Nagarro and Amazon served as the empirical basis for identifying the most vulnerable points and empirically validating the solutions that formed the substantive core of the principles proposed in this article.

Case analysis: designing an Ad Framework in the Amazon ecosystem. The main task was to design a new frontend architecture for an advertising framework that would support the generation of new advertising layouts using AI and ML methods. This task embodies the essence of the problem under study. The existing system exhibited a number of significant limitations:

– Scalability of layouts. Support for only about ~15 standardized ad sizes, which directly limited monetization potential.

– Performance. A monolithic build configuration (webpack) led to increased First Contentful Paint (FCP) time and significant client-side latency.

– AI integration. The architecture was initially not intended for dynamic UI generation; ad layouts were hard-coded in the codebase.

To overcome these limitations, a comprehensive reconfiguration of the architecture was carried out, based on the principles identified in the course of the theoretical and methodological analysis of the literature.

Before integrating resource-intensive AI processes, it was necessary to eliminate the basic performance issues. Modern build optimization techniques were implemented [7, 8]:

1. Tree Shaking. Aggressive elimination of unused code from the final bundles.
2. Lazy Loading. Asynchronous loading of modules that are not critical for the initial rendering (for example, complex widgets and external scripts).
3. Server-Side Rendering (SSR). Rendering of critical modules and the base layout on the server side to accelerate FCP [9, 10].

The set of measures listed above made it possible to reduce client-side latency by approximately 30 %. This created the required performance margin for the subsequent integration of AI modules without degradation of the user experience.

Architectural support for dynamic user interface generation. The most nontrivial task was to support layout generation by means of AI. A naive strategy would assume direct HTML/CSS generation by the AI model, which is an architecturally fragile and unsafe solution. Instead, a component-oriented architecture was implemented. The operation of the updated architecture is organized as follows:

– The AI service does not construct the UI directly; instead, it produces a declarative JSON structure describing the composition and configuration of the components used.

– The frontend application based on React receives this JSON through an API gateway [10].

– A dedicated rendering engine on the frontend side recursively traverses the JSON tree and dynamically mounts the corresponding React components from the design system library.

Such an architectural scheme, based on the ideas of generative design [3], provided a qualitatively different solution to the scaling problem: the system began to support more than 12 000 variants of ad formats instead of the previous 15, since the AI obtained the ability to arbitrarily combine components in any permissible container.

Decomposition and scalability of the system. To manage the increased architectural and organizational complexity, a new design system was developed, engineered to be reusable, adaptive, and scalable. In essence, such a design system represents a variety of micro frontend [9], within which each component (button, banner, form, etc.) is designed, developed, and versioned autonomously. The high adoption rate — about 40 % of teams in the organization within the first six months — serves as empirical confirmation of the effectiveness of the decomposed approach. This made it possible for different teams, including teams working on AI components, to evolve the functionality of their subsystems independently while relying on a shared yet flexibly configurable UI component library.

Integration challenges and ensuring accessibility. In the course of implementation, practical challenges were also identified, in particular the difficulties of integration with legacy systems and the requirements for data quality for AI models. Additionally, when generating thousands of layout variations, the issue of digital accessibility (a11y) became particularly acute. Generative UI is capable of producing layouts that turn out to be visually unreadable, insufficiently contrasted, or hard to operate using the keyboard.

**These risks required:**

– inclusion of automated a11y tests in the CI/CD pipeline (drawing on practices established at Nagarro);

– designing the components of the design system in strict accordance with WAI-ARIA principles.

## 4. Discussion

Analysis of the scientific literature, as well as the practical implementation data presented in the Results section, makes it possible to move from a descriptive fixation of existing solutions to the construction of a synthetic architectural model. Modern approaches such as micro-frontends [6, 9] and server-side rendering (SSR) patterns largely address the challenges of development scalability and performance optimization; however, they remain fragmentary with respect to a unified architectural foundation for integration with AI services that perform dynamic control of the user interface.

As a result of the conducted analysis, an original conceptual model is formulated: The Modular AI-Integrated Frontend Architecture (MAI-FA). This architecture assumes the decomposition of the frontend application into three loosely coupled layers that interact with each other through strictly defined APIs and formalized data flows, which are presented for greater clarity in Figure 1.

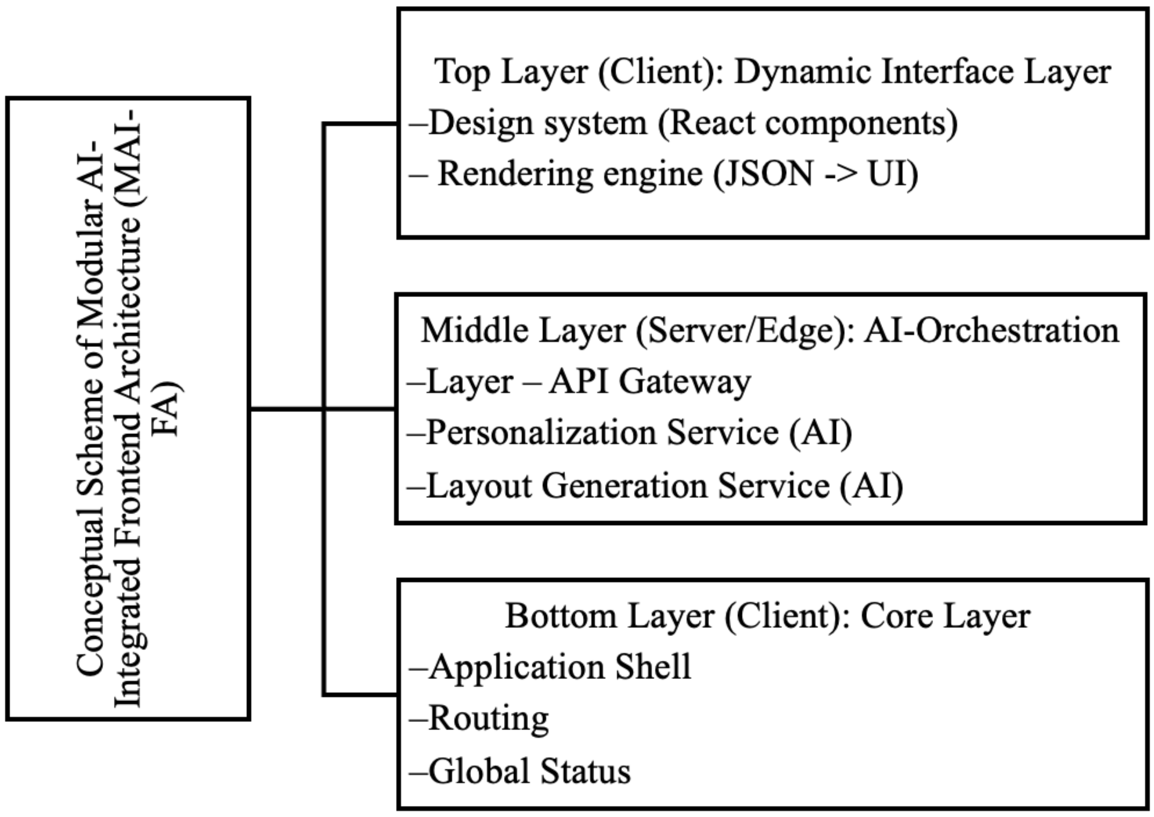
Conceptual Scheme of Modular AI-Integrated Frontend Architecture (MAI-FA)
Top Layer (Client): Dynamic Interface Layer
–Design system (React components)
– Rendering engine (JSON -> UI)
Middle Layer (Server/Edge): AI-Orchestration
–Layer – API Gateway
–Personalization Service (AI)
–Layout Generation Service (AI)
Bottom Layer (Client): Core Layer
–Application Shell
–Routing
–Global Status

**Fig.1. Conceptual scheme of Modular AI-Integrated Frontend Architecture (MAI-FA) [6, 9]**

The Core Layer represents a basic shell application that encapsulates fundamental system functions: authentication, routing, global state management, and module initialization/loading.

The Dynamic Interface Layer includes a design system in the form of a library of atomic components and a specialized Rendering Engine. The functional purpose of this layer is reduced to the interpretation and visualization of the declarative interface descriptions provided to it; it does not perform content selection and is responsible exclusively for how this content is rendered.

The AI Orchestration Layer is implemented as an intermediate layer (as a rule, on the server or at the level of Edge infrastructure), acting as a facade between the frontend and the set of ML models. Its task is to transform business requests into declarative JSON structures suitable for direct consumption by the interface layer and subsequent materialization in the form of the user UI.

The proposed approach addresses the key problems identified in the course of the conducted analysis. The practical experience of developing a framework for more than 12,000 advertising creatives can be interpreted as a concrete materialization of the Dynamic Interface Layer operating under the control of the AI Orchestration Layer. Subsequently, Table 1 provides a comparison of MAI-FA with the traditional monolithic approach in the context of AI component integration.

***Table 1. Comparative analysis of the Traditional monolithic architecture and MAI-FA***

| Parameter | Traditional Monolith (CSR/SPA) | MAI-FA (Proposed model) |
|---|---|---|
| UI management | Hard-coded in components. Logic and presentation are mixed. | Declarative. The UI is controlled by a JSON structure from the AI service. |
| Scalability | Low. Adding a new layout requires rebuilding and releasing the frontend. | High. New layouts are created by the AI service without modifying the frontend code. |
| Performance | Risk of bundle bloat. Requests to the AI may block rendering. | Optimized. The core loads quickly (SSR/Islands), the UI is generated asynchronously. |
| AI integration | Direct API calls from components. The frontend is aware of the models. | Abstracted. The frontend interacts only with the Orchestration Layer. |
| Team independence | Low. Frontend and AI teams are tightly coupled. | High. Teams are separated by an API contract (JSON). |

The proposed architecture is based on the following key principles.

Principle 1: UI decomposition. The frontend layer must not encapsulate the business logic of forming the interface layout; its role is reduced to functioning as a dumb renderer. Such isolation is achieved through the use of the micro-frontend pattern [9] at the design system level, which creates a technical foundation for an AI service that assembles the interface from pre-prepared and independently deployable component bricks. The author's practical experience in building a design system adopted by 40% of product teams empirically confirms the effectiveness of this principle. Below, Figure 2 presents a description of the UI decomposition scheme based on micro-frontends.

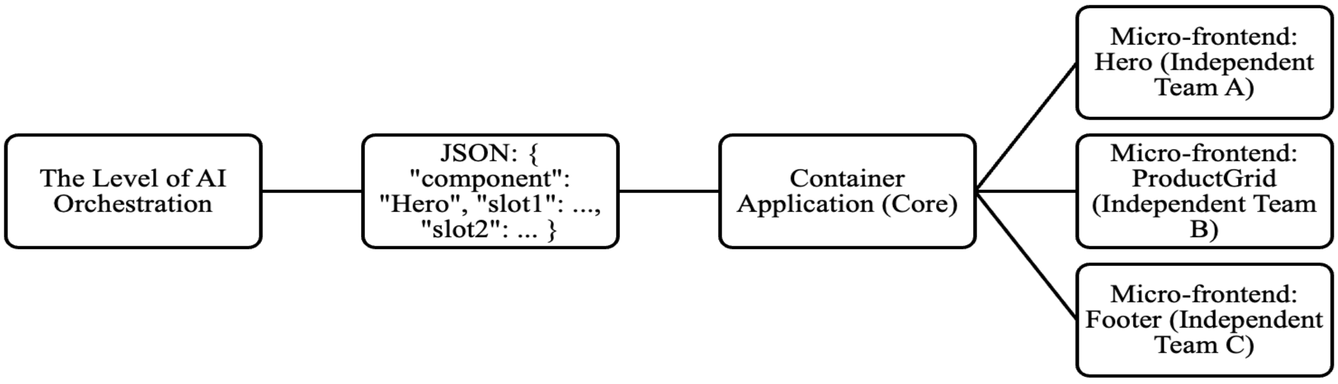
The Level of AI Orchestration
JSON: { "component": "Hero", "slot1": ..., "slot2": ... }
Container Application (Core)
Micro-frontend: Hero (Independent Team A)
Micro-frontend: ProductGrid (Independent Team B)
Micro-frontend: Footer (Independent Team C)

**Fig.2. UI decomposition scheme based on micro frontends [9]**

The next principle is aggressive performance optimization at the build level. The integration of AI functionality inevitably leads to an increase in the volume of client-side JavaScript code (additional libraries for data processing, telemetry, analytics) as well as to a higher number of network requests. In the absence of targeted optimization, this results in the deterioration of key performance indicators and the degradation of Core Web Vitals. A 30% latency reduction effect is achieved through the combined application of Tree Shaking, lazy loading, and server-side rendering (SSR). Modern build tools (webpack, Rspack) and frameworks (Next.js, Astro) provide advanced mechanisms for implementing these approaches, including architectural patterns at the level of the Islands Architecture, which make it possible to minimize redundant client-side initialization.

Principle 3: API-first and contract-based design. The primary mediator between the frontend layer and the AI system is the API. Within the MAI-FA model, this role of a contract interface is performed by a JSON schema for the declarative description of the user interface. Such a contract is required to have a strict versioning system and formalized documentation (in particular, based on JSON Schema), which ensures the robustness of integrations and the controlled evolution of interaction protocols. Such a contract specification creates the conditions for parallel and loosely coupled work of AI teams and frontend teams without violating the integrity of the architecture.

Principle 4: Built-in accessibility. In a configuration where the generation of user interfaces is delegated to an AI model, there arises a significant risk of forming interfaces that do not meet accessibility requirements [1, 5]. The Accessibility-by-Design principle assumes that responsibility for compliance with accessibility standards is shifted from the stochastic AI model to the deterministic level of the design system (Principle 1). The basic components must be initially constructed as accessible, with correct support for WAI-ARIA, focus management, adherence to contrast ratios, and other regulatory criteria. In such an architecture, the task of the AI is reduced to composing pre-accessible component elements, which minimizes the likelihood of systematic violations of inclusive design requirements and makes the behavior of the system predictable from the perspective of accessibility.

For the subsequent assessment of the effectiveness of implementing the described architectural model, it is advisable to use the system of key performance indicators (KPI) described in Table 2.

***Table 2. KPIs for assessing the effectiveness of MAI-FA [1, 5]***

| Category | KPI | Metric |
|---|---|---|
| Technical | Performance | FCP (First Contentful Paint) / LCP (Largest Contentful Paint) |
| Technical | Performance | AI Orchestrator response latency |
| Technical | Scalability | Core bundle size (Core Layer) |
| Product | Flexibility (Velocity) | Time-to-Market (for a new UI layout) |
| Product | Scalability | Number of supported UI variants (layouts) |
| Product | Quality | Level of Design System adoption |
| Quality | Accessibility (a11y) | % of errors in automated a11y tests |

It is advisable to demonstrate the operation of the model using the example of the end-to-end data flow shown in Figure 3, which ensures the generation of an advertising layout, analogous to the architecture implemented in the author's case study.

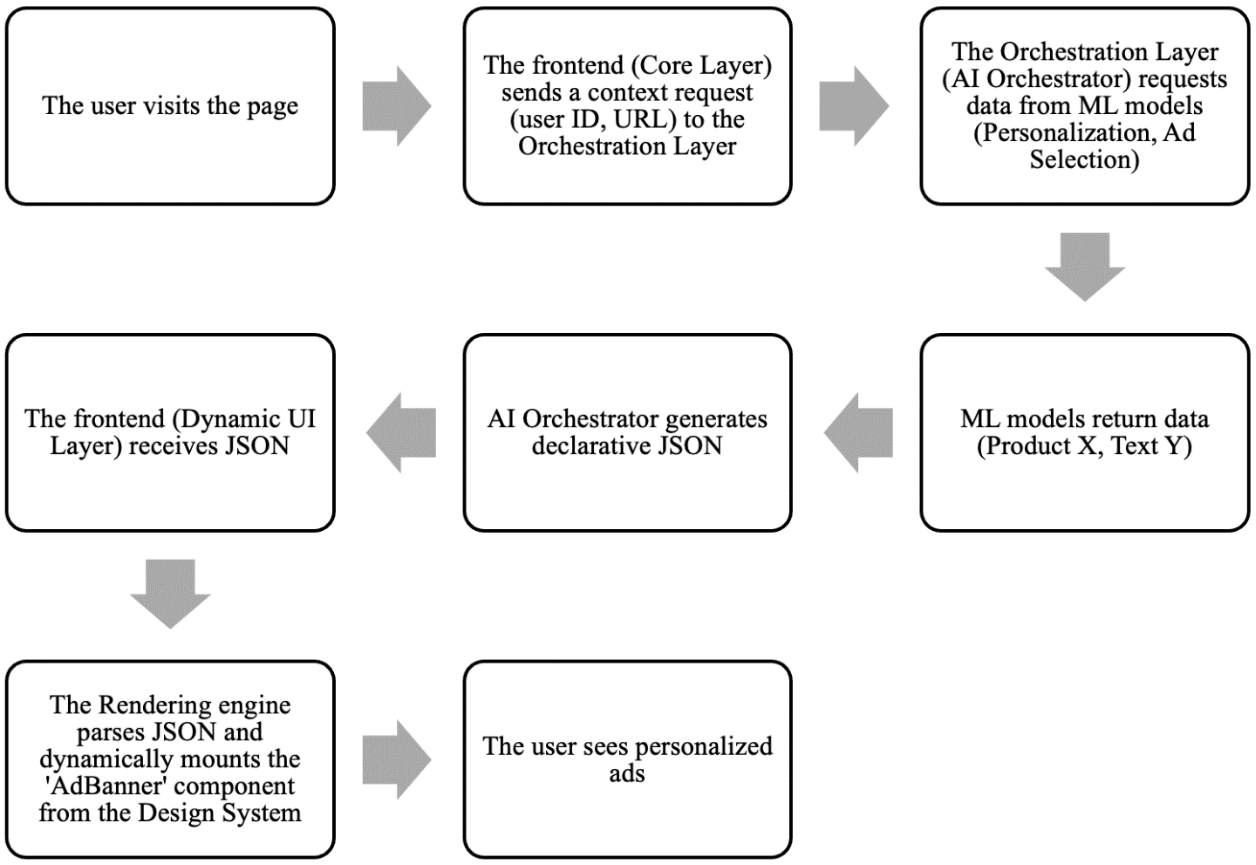
The user visits the page
The frontend (Core Layer) sends a context request (user ID, URL) to the Orchestration Layer
The Orchestration Layer (AI Orchestrator) requests data from ML models (Personalization, Ad Selection)
ML models return data (Product X, Text Y)
AI Orchestrator generates declarative JSON
The frontend (Dynamic UI Layer) receives JSON
The Rendering engine parses JSON and dynamically mounts the 'AdBanner' component from the Design System
The user sees personalized ads

**Fig.3. Data flow during AI-driven UI generation (using AdTech as an example)**

The proposed Modular AI-Integrated Frontend Architecture (MAI-FA) is positioned as a response to the complex challenges arising in the context of contemporary AI-centric web development. The architectural approach systematizes and generalizes practical experience, in particular in solving the problem of large-scale expansion of the number of advertising creatives. MAI-FA reorients the traditional paradigm of the frontend as an API consumer towards a model of the frontend as a high-performance rendering engine for a declarative UI generated by an AI system. This decomposition of responsibility boundaries ensures a synergistic combination of scalability, performance, and flexibility of the development process, enabling the system to evolve without an exponential increase in the complexity of the client side.

## 5. Conclusion

As a result of the conducted study, the stated objective has been achieved: the principles for designing scalable frontend architectures focused on integration with artificial intelligence systems and optimized for such use cases have been formulated and theoretically substantiated.

The analysis of scientific and applied literature made it possible to identify the main classes of problems: increased latency and complexity of state management under conditions of asynchronous AI responses, significant integration barriers with legacy infrastructure, vulnerability to performance degradation under increasing load, as well as the need to introduce new architectural paradigms, including micro frontends and hybrid rendering models.

The systematization of existing approaches, carried out on the basis of a combined analysis of theoretical sources and a practical case study, made it possible to demonstrate that none of the approaches considered, when applied in isolation, provides a solution to the complex task of constructing a robust, high-load frontend with tight AI integration.

The development of the author's model resulted in the construction of the Modular AI-Integrated Frontend Architecture (MAI-FA). The model presented in the Discussion section assumes the decomposition of the system into three interconnected layers and relies on four fundamental principles: decomposition of the UI into isolated components, optimization strictly oriented towards performance metrics, API-first design, and accessibility (a11y) embedded from the outset as a mandatory non-functional requirement.

## References

**1.** Tkachenko, O., Chechet, A., Chernykh, M., Bunas, S., & Jatkiewicz, P. (2025). Scalable Front-End Architecture: Building for Growth and Sustainability. Informatica, 49(1), 137-150.

**2.** Shukla, A. (2025). System design for AI engineering: Adaptive architectures for real-world scalable AI applications. International Journal of Computer Applications, 187 (25), 34-39.

**3.** Yiannoudes, S. (2025). Shaping Architecture with Generative Artificial Intelligence: Deep Learning Models in Architectural Design Workflow. Architecture, 5(4), 94.

**4.** Moussaoui, J.-E., Kmiti, M., El Gholami, K., & Maleh, Y. (2025). A Systematic Review on Hybrid AI Models Integrating Machine Learning and Federated Learning. Journal of Cybersecurity and Privacy, 5(3), 41.

**5.** Palli, S. H. (2023). Design Patterns and Performance Strategies for Scalable Frontend Applications, 9 (4), 676-684.

**6.** Jain, S. (2020). Synergizing Advanced Cloud Architectures with Artificial Intelligence: A Paradigm for Scalable Intelligence and Next-Generation Applications. Technix International Journal for Engineering Research, 7 (3), 1-12.

**7.** Vepsäläinen, J., Hellas, A., & Vuorimaa, P. (2024). Overview of web application performance optimization techniques. International Conference on Web Information Systems and Technologies. Cham: Springer Nature Switzerland, 1-19.

**8.** Prakash Mathew. (2025). Front-End Performance Optimization for Next-Generation Digital Services. Journal of Computer Science and Technology Studies, 7(4), 993-1000.

**9.** Prajwal, Y., Parekh, J. V., & Shettar, R. (2021). A brief review of micro-frontends. United International Journal for Research and Technology, 2(8), 123-126.

**10.** Morozova, O.I. (2025). The future of AI-powered digital accessibility. Aubergine Solutions. Retrieved from: https://www.aubergine.co/insights/the-future-of-ai-powered-digital-accessibility (date of access: 13.10.2025).