DOI: https://doi.org/10.36719/2789-6919/57/205-211

Ilkin Hajiyev
Azerbaijan State Economic University
Master's student
https://orcid.org/0009-0006-6234-4708
ilkinhaciyev955@gmail.com

Design and Research of a React JS-based Multi-Vendor E-Commerce Platform

Abstract

This paper investigates the design, implementation, and architectural evaluation of a multi-vendor e-commerce platform developed using React.js. The study is driven by the growing need for scalable and maintainable digital marketplace systems capable of supporting multiple vendors, dynamic product catalogs, and interactive user experiences within a unified environment. The research focuses on the role of modern frontend technologies in improving system performance, modularity, and usability. In particular, the impact of component-based architecture, centralized state management, client-side routing, and RESTful API integration is examined from both structural and functional perspectives. A layered client-server architecture is proposed, consisting of presentation, business logic, and data management layers, in order to ensure separation of concerns, extensibility, and efficient communication between system modules. The functional responsibilities of administrators, vendors, and customers are clearly distinguished, and the interaction between major software components is analyzed in terms of interoperability and scalability. To strengthen the methodological basis of the study, formalized performance indicators related to response time, interface responsiveness, and data consistency are introduced. The results of the architectural analysis indicate that React.js provides a suitable technological foundation for the development of modular and scalable marketplace platforms due to its reusable component model, Virtual DOM optimization, and compatibility with contemporary frontend development ecosystems. The findings offer both practical and methodological guidance for future web-based commercial platform design.

Keywords: *multi-vendor e-commerce, React.js, marketplace architecture, frontend development, REST API, Redux, scalability, user interface performance.*

İlkin Hacıyev
Azərbaycan Dövlət İqtisad Universiteti
magistrant
https://orcid.org/0009-0006-6234-4708
ilkinhaciyev955@gmail.com

React JS əsaslı Multi-Vendor Elektron Ticarət Platformasının Dizaynı və Tədqiqatı

Xülasə

Bu məqalədə React.js istifadə edilməklə hazırlanmış çoxsatıcılı elektron ticarət platformasının layihələndirilməsi, həyata keçirilməsi və arxitektura baxımından qiymətləndirilməsi tədqiq edilir. Tədqiqat vahid mühit daxilində çoxsaylı satıcıları, dinamik məhsul kataloqlarını və interaktiv istifadəçi təcrübəsini dəstəkləyə bilən miqyaslana bilən və texniki baxımdan müşayiət oluna bilən rəqəmsal marketplace sistemlərinə artan tələbatdan irəli gəlir. Araşdırma müasir frontend texnologiyalarının sistemin performansının, modulluğunun və istifadəyə yararlılığının artırılmasındakı roluna yönəlmişdir.

Xüsusilə komponent əsaslı arxitekturanın, mərkəzləşdirilmiş vəziyyət idarəetməsinin, müştəri tərəfli marşrutlaşdırmanın və RESTful API inteqrasiyasının təsiri həm struktur, həm də funksional baxımdan araşdırılır. Sistem modulları arasında məsuliyyət bölgüsünü, genişlənə bilməni və səmərəli əlaqəni təmin etmək məqsədilə təqdimat, biznes məntiqi və verilənlərin idarə olunması səviyyələrindən ibarət çoxlaylı client-server arxitekturası təklif olunur. Administratorların, satıcıların və müştərilərin funksional vəzifələri aydın şəkildə fərqləndirilir, əsas proqram komponentləri arasındakı qarşılıqlı əlaqə isə qarşılıqlı işləkliyə və miqyaslana bilməyə görə təhlil edilir. Tədqiqatın metodoloji əsasını gücləndirmək üçün cavab müddəti, interfeys reaksiyası və məlumat ardıcıllığı ilə bağlı formallaşdırılmış performans göstəriciləri təqdim edilir. Arxitektura təhlilinin nəticələri göstərir ki, React.js yenidən istifadə oluna bilən komponent modeli, Virtual DOM optimallaşdırılması və müasir frontend inkişaf ekosistemləri ilə uyğunluğu sayəsində modullu və miqyaslana bilən marketplace platformalarının hazırlanması üçün uyğun texnoloji əsas yaradır. Alınmış nəticələr gələcək veb əsaslı kommersiya platformalarının layihələndirilməsi üçün həm praktik, həm də metodoloji istiqamətverici rol oynayır.

Açar sözlər: *çoxsatıcılı elektron ticarət, React.js, marketplace arxitekturası, frontend inkişafı, REST API, Redux, miqyaslana bilmə, istifadəçi interfeysinin performansı*

Introduction

In recent years, advances in information and communication technologies have fundamentally reshaped the organization of trade and service activities. Conventional business models are increasingly giving way to digital platforms that enable broader market reach, faster transaction cycles, and more efficient interaction between suppliers and consumers. Within this transformation, e-commerce platforms have emerged as a core element of the digital economy by providing flexible and scalable environments for commercial operations. Among the major forms of digital commerce, multi-vendor marketplace systems are particularly significant because they allow multiple independent sellers to operate within a unified platform while supporting diverse products, services, and user interactions.

Compared with single-vendor e-commerce systems, multi-vendor platforms require a more sophisticated architectural framework. Such systems must accommodate multiple user categories, including administrators, vendors, and customers, while simultaneously ensuring stable performance, secure data exchange, and coherent business logic. They also require the efficient management of dynamic product catalogs, user sessions, transaction processes, and communication across multiple software layers. As vendor participation and user activity increase, the architectural quality of the platform becomes a critical determinant of scalability, maintainability, and overall operational efficiency (Kravari & Bassiliades, 2018; Sharma & Lijuan, 2015). At the same time, expectations for modern web applications have become considerably higher. Users increasingly demand responsive interfaces, intuitive navigation, real-time interaction, and seamless transitions between pages and functional modules. For this reason, the development of contemporary marketplace systems depends not only on a reliable backend infrastructure but also on an efficient frontend architecture capable of delivering a high-quality user experience. In this context, frontend technologies have assumed a central role in the design of large-scale web applications (Huang & Wang, 2022; Shehzad et al., 2017; Emmanni, 2024; Fowler et al., 2015).

React.js is widely regarded as one of the most effective frontend libraries for the development of interactive and modular web interfaces. Its component-based programming model makes it possible to divide the user interface into reusable and independent elements, thereby simplifying both system development and long-term maintenance. In addition, the Virtual DOM mechanism improves rendering efficiency by reducing unnecessary updates to the actual Document Object Model and, as a result, enhances interface responsiveness and overall application performance (React, n.d.; Emmanni, 2024). When combined with technologies such as Redux for centralized state management, React Router for client-side navigation, and RESTful APIs for communication between the frontend

and backend layers, React.js provides a strong technological foundation for scalable and maintainable e-commerce systems (React Router, n.d.; React Redux, n.d.; Redux Toolkit, n.d.).

The significance of the present study is associated with the increasing demand for multi-vendor digital commerce platforms capable of supporting high user activity, extensible functionality, and efficient software organization. Although numerous practical solutions for marketplace development have already been introduced, the architectural analysis of React.js-based systems remains an important research direction, particularly in relation to modularity, interoperability, and performance-oriented design. A systematic examination of such platforms contributes to a deeper understanding of how modern frontend ecosystems can improve the structural and functional quality of web-based commercial systems (Fowler et al., 2015; Ferreira et al., 2024).

The aim of this paper is to investigate the architectural design principles and implementation features of a multi-vendor e-commerce platform developed with React.js. The study focuses on the layered organization of the system, the interaction among its major software modules, and the role of frontend technologies in ensuring usability, responsiveness, and scalability. Furthermore, formalized indicators related to response time, interface responsiveness, and data consistency are considered in order to strengthen the methodological basis of the research.

The remainder of the paper is organized as follows. First, the general architectural structure of the proposed system is presented. Next, the principal software modules and user categories are analyzed. Subsequently, performance-related indicators and structural characteristics are discussed from a functional perspective. Finally, the main findings are summarized, and the practical significance of the study for the future development of web-based marketplace systems is outlined.

Research

This study examines the design, modular organization, and architectural evaluation of a multi-vendor e-commerce platform developed with React.js. The primary objective of the research is to analyze the interaction between frontend and backend components in multi-user marketplace systems, with particular emphasis on scalability, interface responsiveness, and data consistency. In contemporary digital commerce environments, modular design, component-based programming, and layered architectural principles are widely regarded as essential for ensuring maintainability, extensibility, and long-term operational stability (Emmanni, 2024; Fowler et al., 2015).

Within the scope of the study, the proposed system was structured as a layered client-server architecture composed of the presentation layer, business logic layer, and data layer. This arrangement supports the separation of concerns, enables the independent development of functional modules, and facilitates future scalability and integration with external services (Fielding, 2000; Kravari & Bassiliades, 2018). The frontend of the platform was implemented using React.js, where a component-based approach was adopted to organize the user interface dynamically. Through this structure, interface elements such as product cards, product lists, order pages, shopping cart modules, user dashboards, and administrative panels can be implemented as reusable software components, thereby improving maintainability, interface consistency, and structural adaptability during system evolution (Emmanni, 2024; Ferreira et al., 2024; React, n.d.).

User roles were also explicitly defined as part of the architectural design of the system. Three principal categories were identified within the platform: administrator, vendor, and customer. The administrator is responsible for overall system supervision, approval of vendor accounts, and monitoring of product-related operations. The vendor manages the store environment, adds products, tracks orders, and updates stock information. The customer interacts with the system through product search, order placement, payment processing, and personal account operations. This functional division reflects a common role-oriented organization in multi-vendor e-commerce systems, where the clear allocation of responsibilities contributes to operational transparency, usability, and more secure process management (Kravari & Bassiliades, 2018; Shehzad et al., 2017).

On the frontend side, client-side routing was implemented through React Router in order to ensure efficient interaction between the user and the system. This approach eliminates full-page reloads during navigation and, as a result, improves application responsiveness and the continuity of user

interaction (React Router, n.d.). At the same time, the adoption of a centralized state management strategy was considered appropriate for handling shared data across different parts of the application. In this context, Redux or a similar state management mechanism can be employed to manage shopping cart data, user sessions, selected products, and different stages of the ordering process. Centralized state control strengthens data synchronization, reduces inconsistencies across interface components, and contributes to more predictable application behavior in complex usage scenarios (React Redux, n.d.; Redux Toolkit, n.d.).

On the backend side, the business logic and data processing functions of the system were organized through REST API-based services. The frontend application communicates with the server by sending HTTP requests, while the server processes these requests and returns responses in JSON format. This approach not only enables the technological independence of system modules, but also simplifies integration with external services and third-party platforms. In particular, user authentication, product management, order processing, and payment-related operations were defined as the core functional services of the backend layer (Fielding, 2000; Express.js, n.d.). In addition, secure token-based authorization mechanisms can be employed to protect communication between the client and server layers and to manage user sessions in a more reliable manner (Jones et al., 2015).

The database model incorporates the principal information structures of the marketplace system. Within the scope of the study, the main data entities were defined as *Users, Vendors, Products, Orders,* and *Payments*. The correct establishment of relationships among these entities is essential for maintaining data integrity, ensuring efficient query execution, and preserving operational consistency. For instance, a single vendor may manage multiple products, a single customer may create multiple orders, and each order may be associated with a corresponding payment record. Such relationships should be explicitly taken into account in the database design in order to ensure a stable and logically coherent system structure (MongoDB, n.d.).

To evaluate the architectural efficiency of the system, several formalized performance indicators were introduced. First, the relative system performance indicator can be expressed as follows:

$$P_{sys} = \frac{N_r}{T_{avg}} \qquad (1)$$

where P_{sys}denotes the relative system performance, $N_r$is the number of processed requests within a given time interval, and $T_{avg}$is the average response time of the requests. According to this expression, the processing of a greater number of requests within a shorter average response time indicates a higher level of system efficiency.

To assess interface responsiveness, the delay between the user request and the visual rendering of the updated interface may be defined as:

$$R_{ui} = t_{render} - t_{request} \qquad (2)$$

where R_{ui}represents the interface response time, $t_{request}$is the moment at which the user request is initiated, and $t_{render}$is the moment at which the updated component is displayed on the screen. This indicator is regarded as one of the principal technical parameters affecting user experience in modern web applications (Huang & Wang, 2022; Shehzad et al., 2017).

Data consistency is also of critical importance in marketplace platforms. In particular, the consistency between shopping cart contents, product quantities, order statuses, and payment records is one of the key factors determining system reliability. The data consistency indicator may be expressed as follows:

$$C_d = \frac{N_c}{N_t} \times 100\% \qquad (3)$$

where C_d denotes the percentage of data consistency, N_c is the number of successfully executed and contradiction-free operations, and N_t is the total number of operations. A higher value of this indicator demonstrates greater reliability of the system in terms of information integrity and transaction coherence.

In order to present the system architecture more clearly, the use of schematic representations and tabular forms was considered appropriate during the study. First, the principal system modules and their functional responsibilities are presented in **Table 1**.

Table 1.
Main system components and functions of the marketplace platform

Component	Primary function	Applied approach or technology
Frontend	Organization of the user interface	React.js, component-based structure
Routing module	Management of navigation between pages	React Router
State management	Management of the global application state	Redux / Context API
Backend API	Execution of business logic and data exchange	REST API
Database	Storage and management of data	Relational or document-oriented database model
Authentication module	User login and authorization control	JWT, session management
Payment module	Processing of payment operations	API integration
Admin panel	System supervision and administrative control	Role-based control panel

As shown in **Table 1**, the system is organized in the form of interconnected modules, each of which performs a specific functional responsibility. This modular approach facilitates software maintenance, simplifies error localization, and supports the future extensibility of the platform. In addition, the separation of concerns between the interface layer, routing mechanisms, state control, backend services, and data storage contributes to greater architectural clarity and system scalability (Fielding, 2000; Fowler et al., 2015; Emmanni, 2024).

Figure 1 illustrates the logical sequence of data flow and interaction among the principal components of the proposed system.

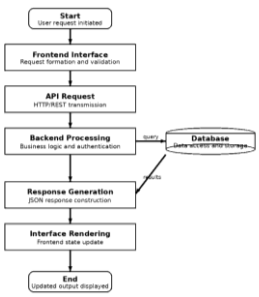

Figure 1. General data flow diagram of the marketplace platform

As shown in Figure 1, every user-generated action is initially formed at the frontend interface, transmitted to the backend through an API request, processed according to the corresponding business rules, and subsequently returned to the interface in a rendered form. This logical sequence ensures the separation of concerns among system modules, improves the transparency of data flow, and supports the analysis of system behavior and functional interaction.

The overall architectural organization of the proposed system can be illustrated in **Figure 2**.

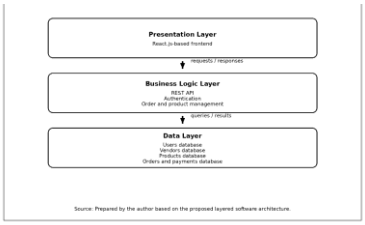

Figure 2. Layered architecture model of the multi-vendor e-commerce platform

The proposed layered architecture is composed of three principal levels: the presentation layer, implemented as a React.js-based frontend responsible for interface rendering and user interaction; the business logic layer, which incorporates REST API services, authentication mechanisms, and product and order management functions; and the data layer, which stores and manages information related to users, vendors, products, orders, and payments. This architectural arrangement indicates that the functional elements of the platform are organized as distinct yet interoperable layers. Such separation enhances architectural clarity, improves maintainability, supports modular extensibility, and contributes to the overall stability and adaptability of the system throughout both development and operational use (Fielding, 2000; Kravari & Bassiliades, 2018).

5trgfFrom a structural perspective, the findings of the study indicate that a React.js-based multi-vendor marketplace platform can be developed as a scalable and user-oriented software system through modular organization, reusable components, client-side routing, and centralized state management. Furthermore, backend connectivity established through REST API services facilitates integration with external services and supports the scalability of the overall software architecture. From this standpoint, the proposed approach may be regarded as methodologically and practically appropriate for the design and implementation of contemporary e-commerce systems (Emmanni, 2024; Fowler et al., 2015; React Router, n.d.; React Redux, n.d.).

Conclusion

1.The study demonstrated that the proposed multi-vendor e-commerce platform can be effectively organized on the basis of a layered client-server architecture consisting of the presentation, business logic, and data layers. This architectural model ensures the separation of concerns among system components and supports a more structured and maintainable software design.

2.The findings confirmed that React JS provides a suitable technological foundation for the development of modular and scalable marketplace systems. Its component-based programming model improves code reusability, interface consistency, and maintainability, while the Virtual DOM mechanism enhances rendering efficiency and interface responsiveness.

3.The use of React Router and centralized state management was shown to improve the continuity and predictability of user interaction within the platform. These technologies contribute to smoother navigation, better synchronization of shared application data, and more stable frontend behavior in complex user scenarios.

4.The backend organization based on REST API services and structured database entities provides an efficient basis for communication, data exchange, and transaction processing. In particular, the functional integration of authentication, product management, order handling, and payment-related processes supports interoperability and system extensibility.

5.The introduction of formalized indicators for system performance, interface responsiveness, and data consistency strengthens the methodological value of the study. These indicators not only support the architectural evaluation of the proposed platform but also provide a useful reference framework for the future design and assessment of similar web-based commercial systems.

References

1. Bilkova, R., & Kopackova, H. (2021). Enhancing E-Commerce by Website Quality. *International Journal of Systems Applications, Engineering & Development, 15*, 99–106. https://doi.org/10.46300/91015.2021.15.14
2. Emmanni, P. S. (2024). Comparative analysis of Angular, React, and Vue.js in single page application development. *International Journal of Science and Research, 12*(6). https://doi.org/10.21275/SR24401230015
3. Express.js. (n.d.). *Express routing*. https://expressjs.com/en/guide/routing.html
4. Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures.* Doctoral dissertation. University of California. https://roy.gbiv.com/pubs/dissertation.pdf
5. Fowler, S., Denuzière, L., & Granicz, A. (2015). *Reactive single-page applications with dynamic dataflow.* Springer. https://doi.org/10.1007/978-3-319-19686-2_5
6. Ferreira, F., Borges, H. S., & Valente, M. T. (2024). *Refactoring react-based Web apps*. *Journal of Systems and Software, 215*, 112105. https://doi.org/10.1016/j.jss.2024.112105
7. Huang, J., & Wang, X. (2022). User experience evaluation of B2C e-commerce websites based on fuzzy information. *Wireless Communications and Mobile Computing, 2022*, Article 6767960. https://doi.org/10.1155/2022/6767960
8. Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT)* (RFC 7519). RFC Editor. https://www.rfc-editor.org/rfc/rfc7519.html
9. Kravari, K., & Bassiliades, N. (2018). *A rule-based eCommerce methodology for the IoT using trustworthy intelligent agents and microservices*. Springer. https://doi.org/10.1007/978-3-319-99906-7_22
10. MongoDB. (n.d.). *Best practices for data modeling in MongoDB*. https://www.mongodb.com/docs/manual/data-modeling/best-practices/
11. MongoDB. (n.d.). *Data modeling in MongoDB*. https://www.mongodb.com/docs/manual/data-modeling/
12. React. (n.d.). *Virtual DOM and internals*. https://legacy.reactjs.org/docs/faq-internals.html
13. React Router. (n.d.). *React Router documentation*. https://reactrouter.com/
14. React Redux. (n.d.). *React Redux documentation*. https://react-redux.js.org/
15. Redux Toolkit. (n.d.). *Redux Toolkit documentation*. https://redux-toolkit.js.org/
16. Sharma, G., & Lijuan, W. (2015). The effects of online service quality of e-commerce websites on user satisfaction. *The Electronic Library, 33*(3), 468–485. https://doi.org/10.1108/EL-10-2013-0193
17. Shehzad, R., Aslam, Z., Ahmad, N., & Iqbal, M. W. (2017). Web usability and user trust on e-commerce websites in Pakistan. *International Journal of Advanced Computer Science and Applications, 8*(12). https://doi.org/10.14569/IJACSA.2017.081267

Received: 05.01.2026
Approved: 12.04.2026