International Journal of Engineering and Information Systems (IJEAIS)
ISSN: 2643-640X
Vol. 10 Issue 10, October - 2026, Pages: 214-219

A Comparative Analysis of Modern Frameworks in Web Application Development: The Case of React, Vue.js, and Angular

Ochilov Baxtiyor Raximovich[1,*], Merajova Shakhlo[2]

[1]*Registan private school, Bukhara, Uzbekistan*
[2]*Scientific Supervisor, PhD, Associate Professor, Bukhara State University, Bukhara, Uzbekistan*
*Corresponding author. Email: ochilovb356@gmail.com

Abstract: *In web application development, React, Vue.js, and Angular are currently the three most widely used JavaScript frameworks, yet the choice of the variant best suited to a given project is often made without systematic criteria, relying instead on personal impression or subjective experience. This article presents a comparative analysis of the three frameworks based on systematic criteria grounded in the ISO/IEC 25010 software quality model (performance efficiency, ease of learning, maintainability, and ecosystem and market-level adoption). The study was conducted by synthesizing official documentation, peer-reviewed scientific articles, university theses, an open benchmark dataset (js-framework-benchmark), and major survey results (State of JS 2024, W3Techs). The results show that in raw performance on DOM operations, Angular leads in most scenarios; React holds an advantage in ease of learning, flexibility, and ecosystem breadth; while Vue.js stands out as a balanced solution for small and medium-sized projects. The article concludes with practical recommendations for students and young developers, grounded in systematic framework-selection criteria.*

Keywords: *web application development; JavaScript frameworks; React; Vue.js; Angular; front-end architecture; comparative analysis; software quality*

1. INTRODUCTION

Over the past fifteen years, web application architecture has undergone a fundamental shift: from traditional multi-page sites that fully reload on every request, toward single-page applications (*SPA*) that update dynamically within the browser. The first broad and systematic scientific analysis of this paradigm was carried out in [1], which demonstrated for the first time that the fundamental problems of testing and analyzing Ajax-based single-page applications (namely, that page state is managed on the client/browser side rather than the server side) render traditional web-testing methods ineffective.

Because of the complexity of building single-page applications, the developer community devised a *component-based* architecture, in which the user interface is divided into independent, reusable blocks. Among the tools implementing this approach, three frameworks continue to lead the market: React [13], maintained by Meta (formerly Facebook); Vue.js [14], a community-governed progressive framework; and Angular [15], a full, TypeScript-based framework developed by Google. As of 2026, React is used on 6.0% of all websites (with an 8.0% market share among JavaScript libraries), Vue.js on 0.6%, and Angular on 0.2% [2]; meanwhile, in the 2024 State of JS survey, React, Vue.js, and Angular took first, second, and third place respectively in current usage, with 8,548, 3,976, and 3,642 respondent votes [3].

At the same time, most existing studies comparing these three frameworks rely on only a single evaluative dimension: either raw performance alone [7], or qualitative aspects alone (ease of learning, ecosystem) [10], leaving developers or students with almost no unified, systematic basis for framework selection. This constitutes the main research gap addressed by this article: a systematic comparative analysis that unites several dimensions (performance, ease of learning, maintainability, and ecosystem) within a single, internationally standardized framework, the ISO/IEC 25010 software quality model [5], has not yet been adequately carried out.

The relevance of the topic is also growing for Uzbekistan: according to the State Statistics Committee, the country's IT services exports rose from 92.6 million USD in the first half of 2022 to 545.3 million USD in January-June 2026, an almost 5.9-fold increase over five years (with an annual growth rate of 72-76% in 2023-2025) [4]. This rapid growth is intensifying demand for front-end developers capable of making well-grounded, modern framework choices, which in turn increases the value of scientific-practical materials that systematically compare frameworks for students and young developers.

The aim of this study is to conduct a comparative analysis of React, Vue.js, and Angular based on a multi-criteria system grounded in the ISO/IEC 25010 quality model, and to develop, on this basis, practical recommendations for framework selection. To achieve this aim, the following tasks were set:

1. identify the key differences among the three frameworks in architecture and design philosophy;

2. compare their performance indicators based on available benchmark data and academic research;
3. analyze differences in ease of learning, ecosystem breadth, and market adoption;
4. develop systematic criteria for framework selection by project type, aimed at students and young developers.

2. METHODOLOGY

This study is qualitative-comparative in nature, based on the systematic analysis and synthesis of secondary sources. The comparison criteria were formed from four quality characteristics selected from the international ISO/IEC 25010:2011 standard, "Systems and software Quality Requirements and Evaluation" [5]: performance efficiency, usability (interpreted here as ease of learning), maintainability, and portability (here extended to mean ecosystem and market adoption). The research process was carried out in four stages (Table 1).

Table 1. Step-by-step methodological design of the study

Stage	Content	Result
I	Review of the frameworks' official documentation and architecture	Classification of architecture and philosophy
II	Formulation of criteria (performance, learning, maintenance, ecosystem) based on ISO/IEC 25010	Four-dimensional evaluation framework
III	Synthesis of benchmark data, scientific articles, and theses	Comparative table by criterion
IV	Generalization of results, development of practical recommendations	Decision-making scheme

The data corpus was drawn from the following types of sources: (1) the official technical documentation of all three frameworks [13,14,15]; (2) peer-reviewed scientific articles and conference proceedings [6,7,12]; (3) university master's/bachelor's theses [10,11]; (4) open benchmark datasets and tools [8,9]; (5) large-scale international surveys and usage statistics [3,2].

As a methodological limitation, it should be specifically noted that this study is based on synthesizing secondary sources and does not include an independent benchmark test conducted by the authors themselves. The performance indicators cited were obtained by different researchers using different framework versions (e.g., Angular 17.1.0, React 18.2.0, Vue 3.3.4 [7]) and different test scenarios, meaning these results may change over time. For this reason, an independent, locally conducted empirical benchmark test is recommended at the end of the article as a direction for future research.

3. RESULTS

3.1. Differences in architecture and design philosophy

An analysis of the official documentation of all three frameworks [13,14,15] reveals fundamental philosophical differences among them (Table 2). React formally calls itself a "library" and covers only the user-interface layer: tasks such as routing and global state management rely on third-party libraries. Vue.js positions itself as a "progressive framework": it can be used as a single script in a small project, or as a complete application together with the official Router and Pinia libraries. Angular, in turn, follows a "batteries included" approach, in which routing, form handling, an HTTP client, and all other necessary tools are built into the framework itself.

Table 2. Comparative architectural description of React, Vue.js, and Angular

Property	React	Vue.js	Angular
Type	Library	Progressive framework	Full framework
Main language	JS/JSX (TS optional)	JS/template (TS optional)	TypeScript (mandatory)
Render model	Virtual DOM, one-way flow	Virtual DOM + reactivity	DOM + signals (v17+)
Maintainer	Meta	Independent team (E. You)	Google
Philosophy	UI layer only; requires third-party libraries	Progressive; optional official add-ons	"Batteries included": everything built in

One important difference lies in the typing system. In Angular, TypeScript is mandatory and serves as the primary language, while in React and Vue.js it is used as an optional add-on. This difference has practical significance: a large-scale empirical study analyzing 604 front-end projects on GitHub (299 in JavaScript, 305 in TypeScript, totaling over 16 million lines of code) found that projects written in TypeScript had significantly fewer "code smells" and lower cognitive complexity per line of code than JavaScript projects, but no statistically significant difference was found in bug-fix ratio or defect-resolution time [12]. This result indicates that Angular's mandatory TypeScript requirement improves code comprehensibility but does not automatically reduce the number of defects.

3.2. Comparative analysis of performance indicators

The most widely used open tool for measuring DOM operation performance is the js-framework-benchmark project, which automates measurement of operations such as row creation, update, and replacement across more than 186 framework implementations using the Chrome browser [9]. Using a similar methodology, Åbo Akademi University's DOM benchmark dataset compared React, Angular, Vue, and Svelte [8].

According to a study conducted on newer versions (Angular 17.1.0, React 18.2.0, Vue 3.3.4) and published in the Journal of Computer Sciences Institute [7], Angular demonstrated the highest performance in five of eight test scenarios, Vue.js led in the 10,000-row table-creation scenario, while React was fastest in the scenarios for clearing the entire table and reversing row order (Table 3). Interestingly, the authors of that study had initially hypothesized that Vue would be fastest, followed by Angular, with React last, but the results did not confirm this hypothesis; instead, Angular secured the overall lead [7]. The authors also found that browser choice significantly affected the results: in a number of scenarios, Firefox performed noticeably faster than Chrome, suggesting that browser choice may influence end-user experience more than framework choice does [7].

Table 3. Comparative results on performance, ease of learning, and code quality

Criterion	React	Vue.js	Angular
DOM performance (8 scenarios)	Fastest in 2/8 (remove, reverse)	Fastest on 10,000-row table; 2nd in the rest	Fastest in 5/8, overall leader
Learning curve	Relatively gentle; requires JSX	Gentlest; close to HTML/CSS/JS	Steepest; requires TS, DI, RxJS knowledge
Load/memory	Good indicators	Average	Heavier, handles large data well
Code quality (with TS)	Fewer "code smells", lower complexity	TS optional, improved in v3	TS mandatory; quality enforced automatically

At the same time, a study based not on raw performance alone but on a multi-criteria decision-making method (Analytic Hierarchy Process, AHP) [6] combined test results across performance, modularity, and usability, and concluded that React is the most suitable option for a simple single-page front-end application. This is important evidence that a choice based on raw benchmark figures and one based on practical project needs do not always lead to the same outcome.

3.3. Ease of learning and developer experience

According to a thesis study that practically compared React and Angular using a Todo List application as an example [10], React has a relatively gentle learning curve and demonstrates better initial load time and memory consumption, and its flexibility and broad ecosystem make it well suited to a variety of projects requiring rapid development. Angular, by contrast, provides a cohesive, structured feature set, handles large volumes of data well, and offers a standardized development environment, making it a better option for complex, enterprise-grade applications [10]. A broader study conducted at the Prague University of Economics and Business [11] compared all three frameworks across numerous criteria (architecture, performance, ecosystem, community support, flexibility, scalability, and labor-market demand) and emphasized that framework choice depends on project requirements, team experience, and organizational goals.

3.4. Ecosystem and market adoption

Statistics from the web-technology monitoring service W3Techs, as of October 22, 2026, show that React is used on 6.0% of all monitored websites (with an 8.0% market share among JavaScript libraries), Vue.js on 0.6% (0.8% market share), and Angular on 0.2% (0.3% market share) [2] (Table 4). These figures cover all open websites and often include older, smaller projects, so they may not fully reflect adoption trends among professional developers.

Table 4. Framework usage on websites and popularity among developers

Indicator	React	Vue.js	Angular
Usage on websites	6.0%	0.6%	0.2%
JS library market share	8.0%	0.8%	0.3%
Currently used (State of JS 2024)	8,548 (1st)	3,976 (2nd)	3,642 (3rd)

Developer survey data complements this picture: in the 2024 State of JS international survey, 12,802 people (91% of respondents) answered the "currently used" question, with React, Vue.js, and Angular taking first, second, and third place with 8,548, 3,976, and 3,642 votes respectively; overall ecosystem satisfaction (happiness) averaged 3.8 out of 4 points among 12,964 respondents (93%) [3]. These results confirm React's clear leadership, Vue.js's stable second place, and Angular's strong position within the enterprise segment

while holding a comparatively narrower share of the overall market.

3.5. Mobile application and server-side rendering (SSR) capabilities

When choosing a modern web project's stack, it is important to consider not only in-browser performance but also a framework's server-side rendering (SSR) capabilities, which matter for mobile platforms and search engines. All three frameworks have their own official or semi-official solutions in this direction. In the React ecosystem, the Next.js framework, maintained by Meta, combines SSR, static site generation, and file-based routing, and also allows native iOS and Android mobile applications to be built from a single codebase via React Native [16]. In the Vue.js ecosystem, the Nuxt framework performs a similar role, layering SSR, automatic routing, and a modular architecture on top of Vue's reactivity system [17]. Angular, meanwhile, provides server-side rendering through an official module called Angular Universal, which makes it even more attractive for large, enterprise applications with strict SEO requirements [15].

This additional dimension further complicates framework selection: the choice is no longer simply "React or Vue or Angular," but rather between full ecosystem combinations such as "React + Next.js," "Vue + Nuxt," or "Angular + Angular Universal." The practical takeaway for students and young developers is that, when learning a framework, understanding not just the core library but also the meta-framework ecosystem built around it (Next.js, Nuxt, Angular Universal) is essential for staying competitive in today's labor market.

4. DISCUSSION

The results obtained show that there is no single, universal answer to which framework is "best": each framework reveals its advantages under particular project types and team conditions. React's main strength lies in its flexibility, the broadest ecosystem, and the largest developer community [3], which makes it attractive for a wide variety of projects, especially those with rapidly changing requirements; at the same time, its "UI layer only" philosophy places many architectural decisions (routing, state management) on the developer, which can create additional complexity for inexperienced teams.

Vue.js, in turn, is especially well suited to small and medium-sized projects, as well as educational purposes, owing to its gentlest learning curve and progressive adoptability [10], a point of particular practical importance for students who are just beginning to learn programming. Its comparatively smaller market share [2] does not reflect lower quality or performance, but rather an ecosystem built around a smaller yet active community with a high satisfaction rate.

Angular, because of its mandatory TypeScript requirement [12] and built-in architectural tools (routing, dependency injection, form handling), remains the most suitable option for enterprise-grade applications developed by large teams and requiring long-term maintenance. Its strong performance on DOM operations [7] further reinforces this advantage, although its steep learning curve can be an obstacle for small teams or short-term projects.

It should be particularly noted that the performance results cited [7] were obtained under one specific set of conditions (particular framework versions, particular test scenarios, a particular browser), while other studies [6] have reached different conclusions. This shows that performance differences among frameworks vary from version to version and depend on measurement methodology, and that a final decision should therefore not be based on a single benchmark result alone but should take the project context into account.

From the standpoint of the Uzbekistan context, the almost sixfold growth in IT services exports over five years [4] is increasing demand for skilled front-end development personnel. Under these conditions, a practical recommendation for educational institutions is that students just beginning to learn programming should start with frameworks that have a relatively gentle learning curve, such as Vue.js or React, while students interested in later working on enterprise projects should be advised to study Angular and TypeScript in depth.

Combining all the criteria discussed above (performance, ease of learning, ecosystem, and SSR/mobile capability), a practical decision-making scheme for framework selection by project type is summarized in Table 5. This scheme should be regarded not as a final, rigid rule but as a starting point: in practice, framework choice also depends on a team's existing experience, a project's long-term maintenance needs, and an organization's technology strategy.

Table 5. Practical scheme for framework selection by project type

Project type	Recommended framework	Main reason
Small/startup project, rapid prototyping	Vue.js or React	Gentle learning curve, fast launch
Enterprise, long-term application	Angular	Mandatory TypeScript, built-in architecture
Mobile application (iOS/Android)	React (React Native)	Native app from a single codebase

Project type	Recommended framework	Main reason
SEO-critical content/e-commerce site	Vue+Nuxt or React+Next.js	Strong SSR and static generation
Educational/learning purposes	Vue.js	Gentlest entry curve, clear syntax

It is also worth noting that competition among frameworks is not static: each new version (for example, Angular's signal-based reactivity system or React's server components) may change some conclusions of earlier analyses. For this reason, the comparative results presented in this article should not be taken as a "frozen truth," but rather as a scientific-practical foundation reflecting the state of affairs in 2024-2026 that will require periodic updating.

5. CONCLUSION AND RECOMMENDATIONS

In this study, a systematic comparative analysis of React, Vue.js, and Angular was conducted across four criteria grounded in the ISO/IEC 25010 quality model [5]: performance, ease of learning, maintainability, and ecosystem/market-level adoption. The results showed that in raw performance on DOM operations, Angular leads in most scenarios [7]; React holds an advantage in ease of learning, flexibility, and the broadest ecosystem/community [3]; while Vue.js, with the gentlest learning curve and progressive adoptability, is a balanced solution for small-to-medium projects.

Based on the study's findings, the following practical recommendations were developed. First, students and young developers who are just beginning to learn programming are advised to start with Vue.js or React, since they have a relatively gentle learning curve. Second, Vue.js or React is a balanced choice for rapid prototyping and medium-sized commercial projects. Third, for long-term enterprise applications maintained by large teams, Angular's mandatory TypeScript requirement and built-in architectural tools provide an advantage. Fourth, framework choice should never be based on a single benchmark result alone, but should be grounded in project requirements, team experience, and long-term maintenance needs.

The main limitation of this study is that it is based on synthesizing secondary sources. For future research, it is recommended that an independent, locally conducted empirical benchmark test be carried out with the participation of students from Uzbekistan's universities, and that a survey on framework-selection criteria be conducted among local developers; this would help produce more precise, context-appropriate recommendations for Uzbekistan's IT education system.

Funding statement: This study did not receive a specific grant from any public, commercial, or non-profit funding organization.

Author contributions: B.R. Ochilov: literature collection and analysis, construction of comparative tables, writing of the initial draft of the manuscript. Sh. Merajova: supervision of research methodology and conceptual framework, manuscript review and editing. Both authors read and approved the final version of the manuscript.

Informed consent statement: Not applicable. This study is theoretical-comparative in nature and does not involve human participants, surveys, or primary data collection.

Data availability statement: No new data were generated in this study. All indicators cited were obtained from open, publicly available sources, which are listed in the References section.

Conflicts of interest: The authors declare no conflict of interest.

REFERENCES

[1] Mesbah, A. (2009). *Analysis and testing of Ajax-based single-page web applications* (Doctoral dissertation, IPA Dissertation Series 2009-08). Delft University of Technology, Delft, The Netherlands. ISBN 978-90-79982-02-8.

[2] W3Techs. (2026). *Usage statistics and market share of JavaScript libraries for websites.* Retrieved October 22, 2026, from https://w3techs.com/technologies/overview/javascript_library

[3] State of JS. (2024). *Front-end frameworks: State of JS 2024.* Retrieved from https://2024.stateofjs.com/en-US/libraries/front-end-frameworks

[4] UzA / State Statistics Committee of the Republic of Uzbekistan. (2026, August 23). *Uzbekistan's IT services exports have grown almost six times over the past five years.* https://uza.uz/ar/posts/uzbekistans-it-services-exports-have-grown-almost-six-times-over-the-past-five-years_898835

[5] ISO/IEC 25010:2011. *Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE): System and software quality models.* International Organization for Standardization.

[6] Ilievska, F., & Gramatikov, S. (2022). Analysis and comparative evaluation of front-end technologies for web application development. In *Proceedings of ICT Innovations 2022.* https://proceedings.ictinnovations.org/2022/paper/575/

[7] Bogusz, D., Ciszewski, P., & Pańczyk, B. (2024). Performance analysis of web application client layer development tools using Angular, React and Vue as

examples. *Journal of Computer Sciences Institute, 32*, 223-230.

[8] Levlin, M. (2020). *DOM benchmark comparison of the front-end JavaScript frameworks React, Angular, Vue, and Svelte* [Dataset]. Åbo Akademi University. https://etsin.fairdata.fi/dataset/f7f83192-cca1-42c0-9132-8be09d761cf2

[9] Krause, S. (n.d.). *js-framework-benchmark: A comparison of the performance of a few popular JavaScript frameworks* [Software repository]. GitHub. Retrieved October 2026, from https://github.com/krausest/js-framework-benchmark

[10] Khati Chhetri, S. (2024). *Comparative study of front-end frameworks: React and Angular* [Bachelor's thesis]. https://urn.fi/URN:NBN:fi:amk-2024080724089

[11] Hassan, A. U. (2024). *Comparative analysis of trending web development frameworks: Angular, React and Vue.js* [Diploma thesis, Prague University of Economics and Business, Faculty of Informatics and Statistics].

[12] Bogner, J., & Merkel, M. (2022). To type or not to type? A systematic comparison of the software quality of JavaScript and TypeScript applications on GitHub. In *Proceedings of the 19th International Conference on Mining Software Repositories (MSR 2022)*. ACM. arXiv:2203.11115.

[13] React. (2026). *React documentation.* Meta Platforms, Inc. Retrieved October 2026, from https://react.dev/

[14] Vue.js. (2026). *Vue.js documentation.* Retrieved October 2026, from https://vuejs.org/

[15] Angular. (2026). *Angular documentation.* Google LLC. Retrieved October 2026, from https://angular.dev/

[16] Next.js. (2026). *Next.js documentation.* Vercel Inc. Retrieved October 2026, from https://nextjs.org/docs

[17] Nuxt. (2026). *Nuxt documentation.* Retrieved October 2026, from https://nuxt.com/docs