

Comparison of JavaScript Frontend Frameworks – Angular, React, and Vue

¹Anes Karić; ²Nermina Durmić

¹²Department of Information Technologies

International Burch University, Faculty of Engineering and Sciences

Sarajevo, Bosnia and Herzegovina

Abstract:- This research aims to evaluate JavaScript frontend frameworks by their performance and popularity to provide practitioners with guidance in making informed decisions when choosing frontend frameworks for their projects. For that purpose, three applications with the same features were built using three best-known JavaScript frontend frameworks: Angular, React, and Vue. Comparative analysis was employed to conduct the framework comparison using the content of the three applications. For performance evaluation, tools like PageSpeed Insights, WebPageTest, GTmetrix, Pingdom, Lighthouse, and Google Performance were used. In addition, the research also offers the overall comparison of JavaScript frameworks including documentation quality, ability to develop mobile applications, learning curve, built-in features, etc. The results indicate that while React is accepted as the most popular frontend framework, Vue is the one that offers the best performance. The research also compares JavaScript frameworks by specific features they offer such as state management and analyses the implications of these features to the development process. According to the results, these features have big implications on the development process and they are very important when we are making decisions on which framework we will choose for some project.

Keywords:- Frontend, JavaScript, Angular, React, Vue.

I. INTRODUCTION

In the rapidly evolving landscape of web development, JavaScript frontend frameworks have become the main building blocks in creating interactive and responsive web applications. These frameworks are designed to speed up and simplify the development process. They offer a structured approach that helps in managing the complexities of modern web applications. Angular, React, and Vue are widely adopted and they completely changed strategies for creating web applications. Because of that, it became essential to understand their underlying mechanics, capabilities, and performance impacts.

The widespread adoption of JavaScript frontend frameworks began in the early 2010s when AngularJS showed up. JavaScript frontend frameworks are very popular today. Indicator of their popularity is their community and ecosystem. The size and activity of a framework's community are strong indicators of its popularity. One more indicator is the job market where we can see that many developers with specific framework skills are in demand.

JavaScript frontend frameworks have a crucial role in web development these days. Despite that fact, we can see a noticeable gap in comprehensive comparative analyses of JS frontend frameworks that compare their performances, popularity, and suitability for small to medium-sized projects. This gap is especially notable for discussions about state management efficiencies, virtual DOM manipulation, and the extension of these frameworks to mobile development.

This research aims to solve that gap by conducting an extensive comparison of popular JavaScript frontend frameworks. It aims to evaluate their performance, feature offerings, and community support to determine the most suitable frameworks for small and medium-sized applications. By doing so, this study seeks to furnish developers and decision-makers with insights that could guide the selection of a framework that aligns best with their project requirements.

This research is structured in a way that can guide the reader easily through a logical progression of ideas and findings. It begins with an Introduction where the context of the research is established. The Literature review and research question come after the introduction section. The literature review and research question are followed by methodology. After this, the Materials and Methods chapter provides a detailed account of the research design, methodologies employed, and the tools used to collect data, ensuring replicability and transparency of the research process. The Results and Discussion section consists of findings and a detailed explanation of what they represent. Finally, the Conclusion synthesizes the insights gained, reflecting on the implications of the findings and suggesting avenues for future research.

II. LITERATURE REVIEW AND RESEARCH QUESTION

A comparison of JS frontend frameworks (Angular, Vue, React) has been analyzed in many books and scientific research projects.

Uzayr et al. [1] compare whether JS frameworks are complete frameworks or libraries. The first step was defining the framework and library. Both framework and library are code that is written by someone else. That code aims to help in solving some common problems. JavaScript libraries can be described as parts of furniture that we use to add style and function to the house that is already built. Frameworks can be described as a template that we can use to construct and build the whole house. The authors explain that JavaScript frameworks are Angular and Vue. React, on the other hand, is a library. If we compare React, which is a library, with Vue and Angular, which are the frameworks, we can see the difference when we set up projects by using these technologies. Angular and Vue projects are bootstrapped with almost every single thing that we will need to produce a complete, large-scale application. Frameworks have all the resources that large-scale applications need such as tools for creating forms, making network requests, and running automated tests.

Gaetano [2] is comparing whether JavaScript frameworks use DOM or Virtual DOM. Before explaining the differences between DOM and Virtual DOM, the author explains that DOM stands for Document Object Model and that DOM can be described as an abstraction of a text that is structured. From the perspective of web developers, this structured code is an HTML code. Web developers call DOM simply HTML DOM. On the other hand, Virtual DOM can be described as an abstraction of the real HTML DOM. Since the HTML DOM is an abstraction, the Virtual DOM is an abstraction of an abstraction. The author explains that React and Vue use Virtual DOM, and Angular does not. Angular uses Incremental DOM. As we can see, none of the frameworks work with DOM directly because manipulating DOM data directly is a very slow process. The author finds that Virtual DOM can be updated without affecting the real HTML DOM. Virtual DOM has all the properties like real HTML DOM, but it cannot be manipulated with screen changes like the real DOM. In conclusion, the Virtual author says that the virtual DOM will create different copies of the real DOM (the previous and the updated one) and figure out what objects have been updated. Virtual DOM will then update that specific object that is changed in real DOM and then changes on real DOM will cause the changes on screen.

One of the most important parts of every JavaScript framework is state management. Blokdyk [3] explains state management in React. The author explains that React alone is not enough to handle most apps. That is the reason why most developers use other state management third-party libraries like Redux. In React we can use React Hooks features to handle the local state of a component. With React Hooks, we can handle the React state in functional components. Before the 16.8 React version release, the only options for handling

the React state were class-based components. There are ten hooks in total, but the most important are use State and use Effect. We can also write our custom, React Hooks. On the other hand, Redux is used for handling the global state of an application. Redux is an independent JS library. We can use it with Angular and Vue but it is designed to work well with React. The author claims that the best option for complex React applications is combining Redux and React Hooks.

Morgan [4] compares state management in React and Angular. The author explains that Angular is a complete framework and that React is just a library. In complex applications in React, we need to use external libraries such as Redux for communication between components. Since Angular is a complete framework, it does not need any external libraries such as Redux to handle communication between components. The author explains that the best option for communication between components in Angular is a shared service. Shared service does not need to be visible for all components. We can make it visible only to the consuming hierarchy of the components. Angular's Dependency Injection gives us the option to create local services, see dependency references, accomplish modularity, decouple, and test parts of the application easily. Instead of shared services, we can also use the NGRX library for state management. NGRX represents a group of Angular libraries that enable us to have one centralized store for state management data which is accessible from any part of the application. NGRX library is inspired by Redux. It was already mentioned in this section that the author said that Angular has built-in options for state management and that a third-party library like NGRX is not necessary. React is not a complete framework, it is just a library and Redux is required for it.

The author in [5] compares if JS frontend frameworks support two-way data binding or not. Two-way data binding is important because it guarantees that when properties in the model get updated, so does the UI. Also, when UI elements are updated, the model will get updated. Angular supports two-way data binding and its syntax is a combination of parentheses and square brackets. Angular also supports the two-way data binding in forms with the ng-model directive. We can use ng-model only in template-driven forms. If we use reactive forms in Angular, the two-way data binding will not be supported. VueJS also supports two-way binding and we use the v-model directive which is similar to Angular's ng-model directive. V-model is a powerful directive that can speed up the development process significantly if used properly. It makes it very easy to synchronize between user input and the data model. React on the other hand does not support the two-way data binding out of the box like Angular and VueJS. If we want to keep in sync view elements with a model in React, we can achieve it in multiple ways. In class-based components, we can use the on Change event in combination with the state. In React Hooks we can also use the on Change event where the syntax is even simpler in comparison with class-based components.

In the study [6] various JavaScript frontend frameworks were compared based on their performance using real-world applications. The author utilized tools like Lighthouse to evaluate performance metrics including First Contentful Paint, Speed Index, Largest Contentful Paint, Time to Interactive, Total Blocking Time, and Cumulative Layout Shift. Lighthouse generates a Final Performance Score ranging from 0 to 100, with the most crucial metrics being Largest Contentful Paint and Total Blocking Time. Furthermore, the author compared the frontend frameworks in terms of package size and memory usage. Svelte consistently outperformed other frameworks in metrics such as First Contentful Paint Time, Speed Index, Largest Contentful Paint, and Time to Interactive. Vue also demonstrated strong performance across these metrics, while Angular and React generally showed lower scores. In summary, Svelte and Vue emerged as the top performers in terms of performance metrics such as First Contentful Paint, Speed Index, Largest Contentful Paint, and Time to Interactive, as evidenced by the results presented in the study.

The objective of this study is to distill these comparisons into actionable insights, guiding developers and organizations in making informed decisions when selecting a front-end framework. By delving into performance metrics, assessing community engagement, and evaluating the comprehensive feature set, this research aims to outline a clear, comparative framework that highlights each option's strengths and weaknesses.

➤ *Research Questions:*

- Which JavaScript frontend framework shows the best results for small and medium-sized applications in terms of performance (speed, memory usage)?
- What is the most popular JavaScript Frontend Framework?
- How do specific features such as state management, virtual DOM, two-way data binding, and mobile development capabilities compare across Angular, React, and Vue, and what implications do these features have on the development process?

III. METHODOLOGY

In this research, comparative analysis as a quantitative approach was used to evaluate and compare the performance of three popular JavaScript frontend frameworks: Angular, React, and Vue.js. Comparative analysis falls under the umbrella of descriptive research [7]. Comparative analysis is widely recognized in scientific research as a method for drawing distinctions between different entities by evaluating their characteristics against each other [8]. This approach is commonly seen in fields ranging from social sciences to applied technologies, where researchers aim to understand and articulate the differences or similarities between their subjects of study [9]. In our research, we developed three identical applications using three different JavaScript frontend frameworks: Angular, React, and Vue.js. This part of the research aimed to evaluate and compare the performance of these frameworks in real application scenarios.

A. *Choice of Performance Measurement Tools*

In this study evaluating the performance of frontend frameworks - Angular, React, and Vue.js - the selection of tools was important. Chosen were those widely esteemed and utilized in the industry for web application performance assessment, ensuring the reliability and relevance of the results. The list of tools includes PageSpeed, WebPageTest, GTmetrix, Pingdom, Lighthouse, and Google Performance.

B. *Technical implementation*

- **Angular:** We used Angular's complex ecosystem, including RxJS observables and services for state management of the application. Angular, as a complete framework, offers a rich set of state management features without the need for additional libraries [10].
- **React:** For state management in the React application, Redux was chosen. Redux provides a predictable way of managing the application state through a centralized store, which facilitates application development and testing [11].
- **Vue.js:** In the Vue.js application, Pinia was used for state management, a modern approach to state management that offers simple and efficient integration with the Vue.js ecosystem, with easier syntax and a better development experience compared to previous options like Vuex [12].

C. *Challenges and Solutions*

During development, several challenges were encountered, including aligning the authentication functionality, Stripe integration and other features among different frameworks. These challenges were overcome through detailed planning and the application of best practices specific to each framework, as well as using Firebase to homogenize backend processes. Additionally, challenges associated with different state management approaches in each framework were resolved through careful implementation and configuration of Redux, RxJS, and Pinia, ensuring that the logic and performance of the application were not compromised.

D. *Data Collection*

Performance data for each of the three applications, developed using Angular, Vue.js, and React, were collected using a range of performance measurement tools, including PageSpeed Insights, WebPageTest, GTmetrix, Pingdom, Lighthouse, and Google Performance. Each tool provided various metrics related to performance, such as First Contentful Paint, Speed Index, Largest Contentful Paint, Total Blocking Time, and Overall Performance, etc. The results were then entered into tables for each application and page (Home Page, Login Page, Shop Page, Cart and Checkout Update), allowing for direct performance comparison between frameworks.

The dataset used for performance testing of JS frontend frameworks in this study consists of 300 entries. We get that number if we count the number of metrics for each performance measurement tool and multiply by the number of features tested by that tool and then by three (since we tested three frameworks). Our dataset has one column for measurement tools, one for the metrics, one for the features tested and one for each for React, Angular and Vue scores.

There was no need for a data cleaning process in this dataset because our performance measurement tools provided clean and structured data suitable for immediate analysis. In the results section all these tables are represented separately by tool and feature that is tested. According to (Creswell, 2014), the quality of a dataset for comparative analysis depends on its ability to represent all variables comprehensively, not its size. Our dataset is based on different performance metrics and tools and because of that, it is a good basis for further analysis. In the literature review section, several research papers have been described. Frameworks were also compared by popularity. The dataset for this comparison is simple as it consists of four columns: metric and one for each for React, Angular, and Vue scores. Metrics that are used for comparison are (GitHub stars, contributors, and number of users). All this data was collected from the official GitHub repositories of each framework. We have 9 entries in this dataset. There was no need for a data cleaning process in this dataset.

The data analysis involved several steps:

- **Comparing Individual Metrics:** Each metric was compared manually between frameworks to identify specific performance aspects where one framework stands out over the others.
- **Aggregation of Results:** The overall results were aggregated to obtain a comprehensive performance rating for each framework. This was done by summing the scores manually to show overall performance leadership. Each win in a specific metric was one point for that framework.
- **Visual Representations:** The use of graphs and diagrams enabled visual comparison and easier understanding of the data. For example, bar charts are useful for displaying comparative performance metrics by tools, while pie charts provide insight into the overall dominance of a particular framework across all metrics and features.
- **Analysis by Feature:** An analysis of performance by individual application features (e.g., Home Page, Login Page, etc.) was also conducted, providing insight into how each framework performs in certain scenarios.
- **Based on the collected and analyzed data,** conclusions can be drawn about the overall performance of each framework. For example, Vue.js may stand out as the framework with consistently high performance in most tested metrics, while React shows the second-best results. Angular, although somewhat slower in certain metrics, shows robust performance when all characteristics are considered.

IV. RESULTS AND DISCUSSION

In this section, a comparative analysis of three web applications developed using the Angular, React, and Vue frameworks is presented. Each application has the same features like user authentication (with email/password and Google sign-in options), login, a shop page, a home page, product details, a shopping cart, a checkout process, and Stripe integration for payments. PageSpeed Insights, Pingdom, WebPageTest, GTmetrix, Lighthouse, and the Google Chrome Performance Tab were used for analyzing and comparing the performance of these applications. It also

analyzed whether the technology is a framework or a library, its popularity, learning curve, routing solutions, etc., providing a comprehensive overview of each framework's characteristics.

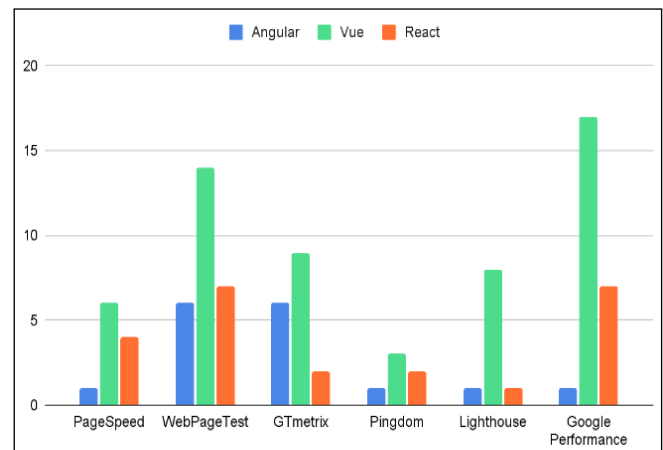


Fig 1:- Top Frameworks Across all Performance Tools

The figure 1 provides a comparative analysis of Angular, Vue, and React applications across various performance benchmarking tools such as PageSpeed, WebPageTest, GTmetrix, Pingdom, Lighthouse, and Google Performance. It aggregates the performance outcomes of multiple pages, including home, shop, and login pages tested with GTmetrix and other tools. These results are answers to research question 1) whose focus was finding a framework that shows the best results for small and medium-sized applications in terms of performance. This synthesis of winners through different performance metrics offers a clear visualization of Vue's superior performance profile in this testing scenario. All these performance tools focus on different metrics and give different results. We can see that even though they measure different metrics, results are constant. Vue is the winner in all performance tools. Vue sometimes won convincingly, sometimes the difference was not huge. React is in second place in all performance tools except GTmetrix. Angular is in third place in all performance tools except GTmetrix and Lighthouse where it shares the second place with React. Each tool has a different set of metrics it prioritizes, such as load time, time to interactive, size of assets, number of requests, etc. Vue's performance profile aligns better with these metrics overall when compared to React and Angular. For PageSpeed Insights and Lighthouse, the most important metrics are metrics related to the user's perceived loading experience, which favors Vue's fast rendering system. WebPageTest focuses on load times and rendering and gives an advantage to Vue because of Vue's efficient rendering and update system. GTmetrix combines results from PageSpeed and YSlow and focuses on frontend structure and performance. Vue has small bundle sizes and this fact gives a big advantage to Vue for the PageSpeed performance tool. Pingdom's focus is on the user centric performance and Vue shows the best results here. Each performance tool focuses on different metrics, and that is the reason why so many different tools were used.

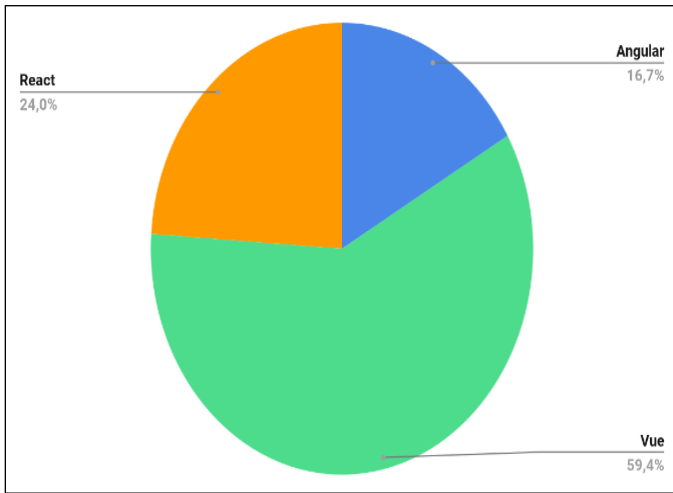


Fig 2:- Top Frameworks Across all Tools and Features

This pie chart provides a percentage-based overview of the performance of Angular, Vue, and React across all metrics in various performance testing tools. Vue leads with a significant margin at 59%, indicating its overall superiority in performance across tools like PageSpeed, WebPageTest, GTmetrix, Pingdom, Lighthouse, and Google Performance. React follows with 24%, while Angular accounts for 17% of the performance share. For performance, a very important factor is file size. Vue offers the possibility to compress code so that it takes less space than the same code written in React.js. Vue applications can have the same features like React applications but at the same time be 30% less in size. This also means that it is possible to create applications in Vue faster because they require less code.

More JavaScript code means that our application will be slower. Vue and React both use Virtual Dom which is great for performance, but React is just a library and we need a lot of additional libraries to complete some basic features. Additional libraries slow down application performance because they require more code to download. Vue has a Vue Router which is the official router for Vue.js and well well-optimized and integrated into the Vue ecosystem. React does not have an official library for routing. Typically a React router is used but it is not well optimized for React like Vue Router for Vue.

Another important feature that Vue offers is state management. Vue has an official state management library which is well optimized only for that framework - Pinia. In React, we typically use Redux but it is not well optimized for React like Pinia for Vue. We also do not need an additional library to handle forms in Vue because Vue offers a V-model for two-way data binding on form inputs.

In React we need additional libraries like Formik. Important features that Vue offers are built-in directives like v-if, v-for, v-bind, v-on, and v-show because they speed the development process and require less code to implement some features. The difference in performance between React and Vue is not very huge but it exists and Vue is the winner in that segment. Angular does not use the Virtual Dom concept, it uses the Incremental Dom concept which is much slower.

Also, Angular is a complete framework. It has a lot of built-in features and it is overkill for small applications. This is why it showed the worst results in most Performance tools.

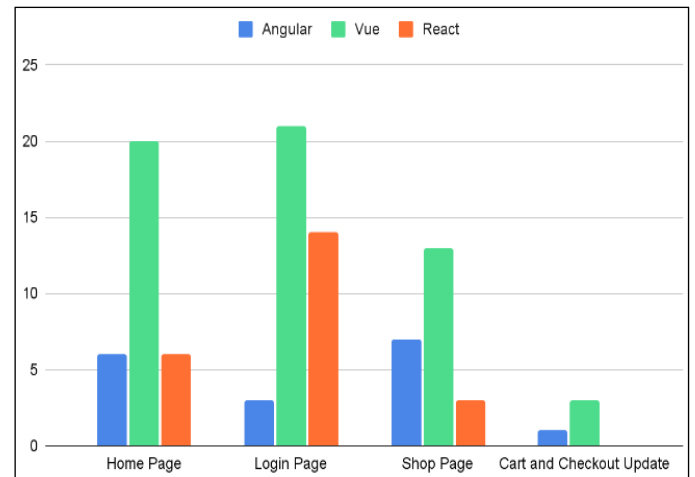


Fig 3:- Top Frameworks Across all Performance Tools per Feature

The bar graph 3 offers a detailed comparison of Angular, Vue, and React's performance across specific features like Home Page, Login Page, Shop Page, and Cart and Checkout Update. Using a range of performance tools, it aggregates metrics to demonstrate each framework's efficiency in real-world applications. As we can see Vue won in all features. In the previous section, it is described in detail why Vue outperforms other frameworks. Vue uses Virtual Dom and is highly optimized out of the box. The overall result in the previous table shows that React is faster than Angular. If we look at the results per feature, we can see that Angular showed better performance results on the Shop Page page than React. It is an unexpected result because React uses the Virtual Dom concept and it is supposed to be faster than Angular. The difference is not huge but Angular showed better performance results in this feature than React. The reason for that can be page content because it has a lot of images loaded.

In the literature review, there were several works where the speed performance of JavaScript frontend frameworks was compared. One of them is [5], where the author compares the performance of React, Vue, Angular, and Svelte with Lighthouse based on their performance using real-world applications. The author utilized tools like Lighthouse to evaluate performance metrics including First Contentful Paint, Speed Index, Largest Contentful Paint, Time to Interactive, Total Blocking Time, and Cumulative Layout Shift. Lighthouse generates a Final Performance Score ranging from 0 to 100, with the most crucial metrics being Largest Contentful Paint and Total Blocking Time.

Furthermore, the author compared the frontend frameworks in terms of package size and memory usage. Svelte consistently outperformed other frameworks in metrics such as First Contentful Paint Time, Speed Index, Largest Contentful Paint, and Time to Interactive. Vue also demonstrated strong performance across these metrics, while Angular and React generally showed lower scores. According

to the author, Svelte and Vue emerged as the top performers in terms of performance metrics such as First Contentful Paint, Speed Index, Largest Contentful Paint, and Time to Interactive, as evidenced by the results presented in the study.

total memory consumption, with Angular being the highest. Angular is a complete framework which has a lot of built in features which lead to higher memory usage. Also Angular’s default change detection strategy is more memory-intensive when compared to Vue and React. Angular also uses powerful services and dependency injection which also consume more memory. These are the reasons why Angular consumes more memory than Vue and React.

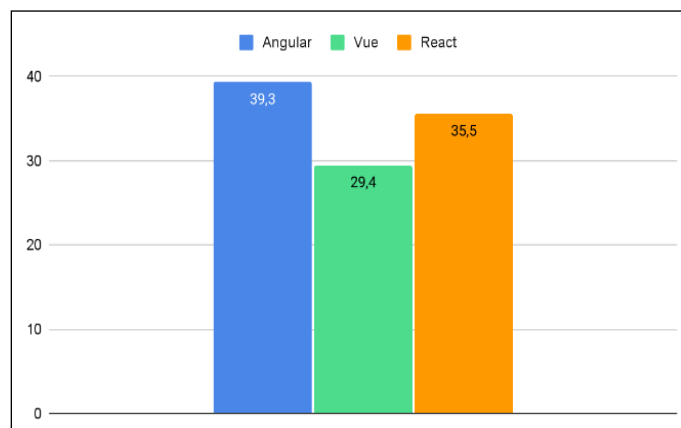


Fig 4:- Graph is Showing Which Framework has the Lowest Memory Usage

The bar graph provides a consolidated view of memory usage across all features for Angular, Vue, and React. Vue emerges as the most efficient, with the lowest aggregate memory usage. In contrast, Angular and React show higher

memory usage. Vue’s reactivity system is fine-grained and precise, which reduces unnecessary work and thus memory consumption. Vue uses a lightweight virtual DOM implementation, which is optimized for memory efficiency. Vue is not a complete framework like Angular, it also requires installing additional libraries but those libraries are mentioned in official Vue documentation and are highly optimized for Vue. React is not a complete framework. It requires installing additional libraries but those libraries are not highly optimized for React like Vue’s libraries. Also to write some feature in Vue that is equal to the feature in React, we need less JavaScript code. More JavaScript code means more memory usage. The difference between React and Vue is not high but it exists and Vue is the winner in this comparison. We can conclude that Vue has the lowest memory usage when compared to React and Angular.

Table 1:- Overall Comparison of Frameworks

Feature	Angular	Vue	React
Type	Framework	Progressive Framework	Library
DOM type	Incremental DOM	Virtual DOM	Virtual DOM
State management	RxJS, NgRx	Vuex/Pinia	Redux, Context API
Release year	2010 (2+ /2016)	2014	2013
Popularity and community	Large and mature	Growing rapidly	Very large and mature
Third party libraries ecosystem	Rich but not so needed	Rich and needed	Very rich and very needed
Data Binding	Two-way	Two-way and One-way	One-way
Folders and Files organization Rules	Very strict	Complete Freedom but with recommendations	Complete Freedom
Mobile Solutions	Ionic	Vue Native, Weex	React Native
Using classes	Mandatory	Optional	Only in legacy code
Language	TypeScript	TypeScript/JavaScript	TypeScript/JavaScript
Component Architecture	Components and services	Single File Components	Components

Table 1. provides a comparative overview of three major front-end technologies: Angular, Vue, and React. Results in this table are derived from a mix of practical applications development, official documentation reviews, and internet research. Results such as type, DOM type, state management, release year, data binding, mobile solutions and its versions migrating easily are derived from official documentation from frameworks. The results displayed in the table regarding the ecosystem of third-party libraries, organization of folders and files, usage of classes, language, component architecture, learning curve, documentation quality and routing were derived from the practical experience of developing identical applications across Angular, Vue, and React frameworks. The ecosystem of third-party libraries for Angular is extensive, yet the comprehensive built-in features often negate the necessity for additional libraries.

Vue's ecosystem, while also rich, leans more on community contributions for advanced capabilities, thus necessitating their usage. React's ecosystem is not just abundant but essential, as it relies heavily on third-party packages for state management and routing, exemplified by Redux and React Router. Folder and file organization in Angular is prescriptive, enforcing a structured approach, while Vue and React offer more freedom, though Vue provides recommendations. Angular mandates TypeScript, enhancing type safety. React and Vue are more flexible, supporting both JavaScript and TypeScript. Component architecture in Angular is service-centric. Vue uses single-file components.

React opts for a compositional model. Learning curves mirror these complexities, with Angular being steeper. Documentation quality varies from Angular's robust but complex to Vue's approachable, and React's practical and community-enriched resources. Routing is built-in for Angular and Vue, while React typically incorporates third-party solutions like React Router. These conclusions are corroborated by practical application development, where these traits directly impact development workflow and efficiency. This hands-on development provided insights into the nuances of each technology, highlighting the differences and similarities that emerge when implementing the same project parameters within each distinct framework environment. Results in table such as popularity and community [13] is derived from internet research. This is also answer on research question 2) about finding the most popular JS frontend framework. Through table 1., we can clearly see differences between JavaScript frontend frameworks in terms of state management, virtual DOM, two way data binding, mobile development capabilities, routing, documentation quality etc.

These results are answers on research question 3) whose focus was on analyzing specific features that JS frontend frameworks offer and defining impacts that they have on development process. These features have big implications on the development process and they are very important when we are making decisions on which framework we will choose for some project.

Angular is a complete framework and if we are working on some enterprise application where extreme performance speed is not required, then Angular is probably the best option. Angular is not using the Virtual Dom concept and that is one reason why React and Vue show better performance in terms of speed. Angular is very strict in terms of folders and files organization and that is good for large enterprise applications because they will have unified architecture across all features.

If we are working on an application that will also have a mobile version that shares the same features as the web version, then React is a great option. React has React Native and they are very similar and a lot of code and logic from the web version can be shared with the mobile version. React is also a very popular framework with a rich library ecosystem and good performance. It also has a better learning curve than Angular. The only problem with React is that it is not a complete framework like Angular and it offers a lot of freedom and flexibility which can cause problems on projects if it is not properly used. This can happen especially if a lot of juniors work on some project without a proper leader.

Vue is not as popular as Angular and React because it is not backed by some large companies like Google or Facebook. Vue uses Virtual Dom like React and shows even better results in terms of performance than React. Vue is a progressive framework and it has more built-in features than React but less than Angular. Vue is probably the best option for small or medium-sized projects that have only web versions and where performance is very important. Vue has

Vue Native which is a similar concept to React Native but is not as popular and powerful as React Native. Vue is also very beginner-friendly and a good option if we have a lot of juniors on some projects because it offers a lot of stuff out of the box such as performance optimization. While it's not possible to definitively declare one framework as the best for all situations, we certainly identified scenarios where a particular framework may be the most advantageous choice.

V. CONCLUSION

The goal of this research was threefold: (1) To find the framework that shows the best results for small and medium-sized applications in terms of performance; (2) To find the most popular JavaScript framework; (3) To define how specific features such as state management, virtual DOM, two-way data binding, and mobile development capabilities compare across Angular, React, and Vue, and what implications these features have on the development process.

The results indicate that Vue shows the best results in terms of performance, while React is found to be the most popular framework. Conclusion for the goal number three is that the outlined features have big implications on the development process and they are very important when we are making decisions on which framework we will choose for some project. Angular is a complete framework and if we are working on some enterprise application where extreme performance speed is not required, then Angular is probably the best option. If we are working on an application that will also have a mobile version that shares the same features as the web version, then React is a great option. React has React Native and they are very similar and a lot of code and logic from the web version can be shared with the mobile version. Vue is not as popular as Angular and React because it is not backed by some large companies like Google or Facebook. Vue is probably the best option for small or medium-sized projects that have only web versions and where performance is very important. Vue is also very beginner-friendly and a good option if we have a lot of juniors on some projects because it offers a lot of stuff out of the box such as performance optimization. While it's not possible to definitively declare one framework as the best for all situations, we certainly identified scenarios where a particular framework may be the most advantageous choice.

The limitation of this research is that its results are mostly based on three smaller applications that were made in Angular, React, and Vue. If those applications were richer with features there is a possibility that performance results could be different. As a future direction, the suggestion is to conduct the same test on more complex applications and draw a conclusion to which extent the complexity and size of applications impact performance-related framework comparison results.

REFERENCES

- [1]. Uzayr, S. B., Cloud, N., and Ambler, T.,“JavaScript Frameworks for Modern Web Development.“ Apress, 2019.
- [2]. Gaetano, C. “How to Choose the Best JavaScript Framework for Your Team“, Mescius, 2019
- [3]. Blokdyk, G, “Comparison Of JavaScript Frameworks“, 5STARCOOKS, 2021
- [4]. Morgan, J., “How To Code In React JS“, Digital Ocean, 2021
- [5]. Wenqing Xu, “Benchmark Comparison of JavaScript Frameworks React, Vue, Angular and Svelte “, University of Dublin, 2021
- [6]. Booth, J. D. “Angular“, Syncfusion, 2019
- [7]. Chanchai Aggarwal, “Descriptive vs Analytical Research“, Shiksha, 2024
- [8]. Jean Kaluza, “What is Comparative analysis“, Dovetail, 2023
- [9]. Appinio research, “What is Comparative analysis and how to conduct it?“, Appinio, 2023
- [10]. G. Geetha, Monisha Mittal “Interpretation and Analysis of Angular Framework“, IEEE, 2022
- [11]. Rohit Jain “React Redux Framework“, Medium, 2022
- [12]. “Vuex official documentation“, <https://vuex.vuejs.org/>, 2024
- [13]. Francesco Borzi “Most Actively Developed frontend framework“, Medium, 2024