Analysis and comparative evaluation of front-end technologies for web application development

Frosina Ilievska and Sasho Gramatikov

Faculty of Computer Science and Engineering Ss. Cyril and Methodius, University, Skopje, North Macedonia
frosina.ilievska@students.finki.ukim.mk

Abstract. In this dynamic, ever-evolving world of web technology, many development tools are created. Everyone can agree that the programming language JavaScript is already in use and will continue to be popular in the future. Despite the many great JavaScript technologies over the past decade, Angular, React.js, and Vue.js remain the most popular.
The construction of a modern single-page application is covered in this study, with an emphasis on the front-end in each of the technologies and the analysis of tests relating to the three key areas of performance, modularity, and usability where data may be evaluated and compared. By analyzing the test findings of the three aspects using the analytical hierarchy process approach, a comparison was produced. This paper provides a response to the question: Which JavaScript framework is best for developing single-page applications in terms of performance, modularity, and usability?
In conclusion, React is the most suitable option for a simple single-page front-end application in our case.

Keywords: front-end comparison, JavaScript technologies, single-page application development, complexity analysis, analytical hierarchy process

1 Introduction

The question “which web development framework should I use?” is one of the questions every developer asks himself before starting a project. Due to the intricacy of the application, manually developing program code may produce inconsistent quality and content. Maintaining applications developed in this way is more complicated. To fix all these problems, developers started using frameworks. The industry for JavaScript technologies has experienced fast growth and change, making it challenging to choose the best framework. Therefore, by creating a simple web application using the three most popular front-end web technologies, we aim to make a thorough analysis and comparison concerning the mentioned three aspects. One of our main goals is to delve deeper into the area of great interest to increase peoples' overall knowledge of JavaScript technologies, so that, in the end, a recommendation can be made as to which technology would be most appropriate for a given project. Another goal is being able to compare technologies and outline their strengths and weaknesses through research and testing. Last but not least, it's crucial to research and understand the distinctions between various technologies in order to make recommendations that can be modified for different users [1][2].

To achieve all this, despite the analysis of the tests to the performance, modularity, and usability, we also use a method called the analytical hierarchy process, which is a method for organizing and analyzing complex decisions, using math and psychologyIn order to make ranking decisions, it combines various metrics into a single total rating, which typically simplifies a decision involving several criteria. We rate relative importance on a scale of 1 to 5, having them in a matrix, and at the end, getting the result for the criteria's importance[1][3][4][5].

$$a_{ij}^{*} = \frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}}$$

Three phases comprise the fundamental process for rating a collection of criteria: developing a pairwise comparison matrix for each criterion, normalizing the matrix, and determining the average value of each row to determine the corresponding rating. Then, in order to analyze several prospective decisions, criteria ratings are applied [6].

The paper aims to indicate how important it is to fully and correctly analyze the process when starting a new project, in order to accurately select a technology. A poorly chosen framework can be a big problem, especially for long-term applications, when software developers realize that the chosen technology does not meet their needs [2][4][5].

The remainder of the paper is structured as follows: We provide a summary of the related work in Section 2. We discuss the phases of comparison in Section 3. In Section 4, we analyze the tests and gather the results from the analysis. In Section 5, the conclusion is made based on the comparison results, also giving the recommendations in which fields this research can be expanded in the future.

2 Related work

Before starting the research for this paper, we went through a lot of resources explaining the same or similar problem. Most of them gave an overview of the three most popular frameworks we also used. [1] gives an overview of multiple page applications. In [2] and [4], the comparison is made on different frameworks such as Dojo, ExtJS, jQuery, Svetle, and Stencil. [7] focuses on the pros and cons these frameworks provide, and in [5] and [8], are explained the main differences in their functionality. [9] gives recommendations for a better performance score, by only comparing the bundle sizes and main characteristics of Angular, React.js, and Vue.js. A single-page application created in AngularJS, with Node.js on the server and MongoDB as the database, is elaborated in [10]. The main point of this book is to learn how to create a web application using these technologies. [11] gives a comparison of Angular, React.js and Vue.js, when creating a complex single-page application.

The main difference with the other resources is the comparison methods and the application complexity. We wanted to show that it doesn’t have to be a complex application, to be able to see the comparison result differences – even on single-page applications, some of the technologies look better than others. Another thing that none of the resources have, is to combine the Lighthouse tool, which is probably the most popular tool for performance testing, with the comparison made with the analytical hierarchy process [11]. Even though there were resources dedicated to the Lighthouse tool [2][12], it is important to mention that in our research paper, we use the newest version, which offers different metrics.

If a developer wants to start a new project, or, recreate an already existing one, this research paper can be a perfect guide, giving him the recommendation of the right technology. It is important to mention that single-page applications can also differ regarding their performance, so why not use the best one?

3 Methodology

To answer the research question mentioned above, 3 technologies are involved. The methodology is divided into three phases: preparation phase, followed by a case study, and finally reaching a conclusion using the analysis phase.

The preparation phase included literary research of the various technologies and services that would be selected for the preparation of this paper. It also included choosing which technologies will be used as well as defining which features of the applications will be compared.

Front-end web applications are created using the frameworks this paper analyzes. The choice of technologies for hypothesis testing was based on how popular the technologies at the time were [7]. The three most widely used technologies, Angular, Vue.js, and React.js, are used to examine the necessary features. A web application may be created as multiple pages (MPA) or only one page (SPA), depending on how they are designed. This research will compare simple single-page applications (SPA).

The next phase is a case study, in which a simple single-page application was developed and implemented with all three selected technologies in parallel. For a feature of the three applications, tests have been performed to understand how the three technologies differ from each other. Once the comparisons were made, the data was collected, and a new comparison was made, until all cases have been completed.

We needed to figure out a way to maintain the bounds of comparing front-end single-page web applications in order to make sure our study was headed in the right direction. This was accomplished using a technique called MoSCoW [11]. This approach helps assignment stays within a range of constraints.

Additionally, when comparing performance, the Lighthouse tool was used, which provides data on several important features: First Contentful Paint (how long it takes for the browser to display the first DOM element's first portion), Speed Index, Largest Contentful Paint (the time a website takes to show the user the largest content on the screen), Cumulative Layout Shift (CLS), Total Blocking Time and Time to Interactive.

Other comparisons of application performance, modularity, and usability can be made manually. In this paper we are analyzing: manipulation with DOM (adding, deleting, and modifying elements), memory allocation (relocating elements), startup time, compile file size (build size), the number of useful NPM packages (rich package ecosystem), as well as the flexibility, reusability of the code, usefulness of the documentation and the learning curve. Equally important features covered by the research are the bundle size, the use of components and their syntax, and the popularity of the frameworks.

Once all the comparisons have been made, we've come to the analysis phase. This stage involves taking the case study test findings that have previously been gathered and using a structured approach classified as the analytical hierarchy process to examine the data and answer to the research problem [1][2][4][5][7][10][11].

4 Results

In this section, we will look at the comparison tests that were made on the single-page application. The application is simple, having the options to add, delete and update an item.

Performance refers to the speed at which content is shown to the user, the amount of time it takes to allocate memory, the amount of time it takes for it to fully load, and finally, the size of the project that was developed. An application was created in the frameworks where the criteria could be measured to collect data. Performance was measured by how quickly the frameworks handled various DOM changes and how quickly they moved memory that had been allocated.

The degree to which the given framework has a robust module ecosystem, the degree to which the code is reusable, and the degree to which the application is flexible may all be indicators of modularity.

The usability aspect revolves around the developer's perspective. Usability could be analyzed by measuring the quality of the documentation used for creating these applications. By creating an application, researching these criteria, and attempting to form educated conclusions, the measurement was carried out [2][4][5][9][10][11][12].

4.1 Performance - Lighthouse tool

JS bundle size

JavaScript bundle size is the only resource that differs for the application in each framework. Since we develop applications with the same look and functionality, the images, CSS, and other resources are the same.

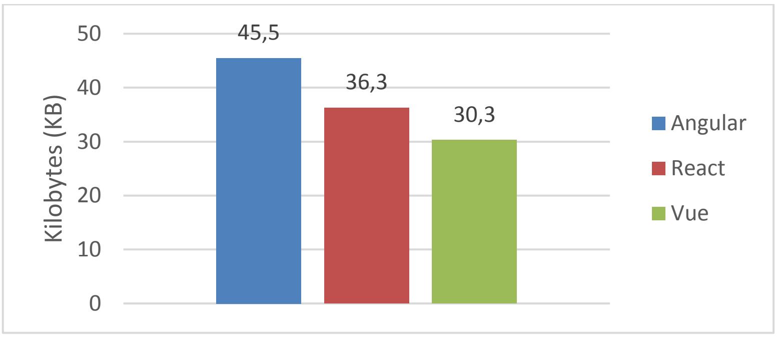

Fig. 1. Bundle Size

As shown above (Fig. 1), Angular’s bundle size is bigger than the other two, but even React and Vue’s bundle sizes are close. With Vue having the less size, Vue is the winner regarding this measurement. The size of Vue apps is extremely small as the framework itself is very lightweight. Developers can break the code into smaller parts with lazy loading components and optimize the load time.

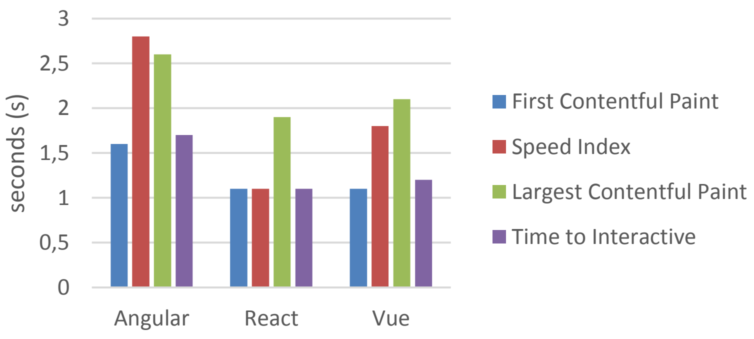

Fig. 2. Performance comparison using the Lighthouse tool

We can see that Angular needs more time than React and Vue regarding all four metrics shown above (Fig. 2), having a large difference, especially in the Speed index and Largest Contentful Paint metrics. React and Vue don’t differ a lot comparing the First Contentful Paint and Time to Interactive metrics, but the Speed index of React is what made him the winner in this performance comparison using the Lighthouse tool. Its component-based approach gives you more speed and flexibility when building web applications.

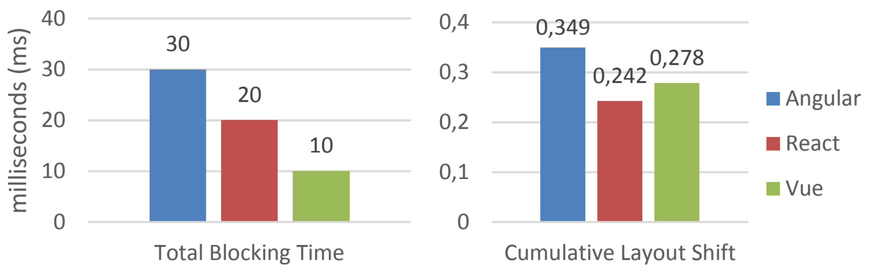

Fig. 3. Total Blocking Time and Cumulative Layout Shift charts

The second part of this comparison confirms Angular’s bad performance score. Unlike before, now Vue has the best Total Blocking Time, and React remains in the first place regarding the Cumulative Layout Shift. Vue uses templates, which makes the process simpler and faster (Fig. 3).

The Lighthouse tool also gives the overall performance score, which in our case, doesn’t have a significant difference – React is in the first place with a score of 92, second is Vue with score of 91 and in the third place we have Angular with 86. Even though the score is almost identical, the Lighthouse tool can help the developers by showing them the big picture of the pros and cons regarding the performance of these frameworks.

4.2 Performance, Modularity & Usability

In this section, we will review all the test results we got from our research. Using the AHP analysis, the three frameworks received points from 1 (worst) to 5 (best). If the difference is very small, it means that there are variations due to several circumstances, so in that case, the points were given equally (Table 1.) [2][4][5][11][13].

Table 1. Average score of the frameworks with and without weights

Tests	Angular	React	Vue	Weight	Angular	React	Vue
DOM-Manipulation	4	3	2	4,7%	0,188	0,141	0,094
Memory Allocation	3	3	3	21,1%	0,634	0,634	0,634
Build Size	4	4	4	3,1%	0,124	0,124	0,124
Startup Time	3	5	3	12,5%	0,376	0,625	0,376
Rich package ecosystem	3	4	3	12,5%	0,376	0,502	0,376
Flexibility	3	4	4	7,6%	0,227	0,303	0,303
Reusability	4	4	4	4,7%	0,188	0,188	0,188
Documentation	3	4	4	21,1%	0,634	0,845	0,845
Learning curve	3	3	4	12,5%	0,376	0,376	0,502
AVERAGE SCORE	3,33	3,79	3,44	100%	3,12	3,74	3,44

Performance

DOM-Manipulation

The test for this part was made by adding, deleting, and editing rows in the application. First, movies were added to the array, then some of them got edited, and lastly, all of them got removed. We used Google Chrome's runtime profiler to gather the results. The test was repeated a few times on different array size. The number of items tested was 100, 500, 1000, 1500, and 2000 and the tests were conducted for all the features.

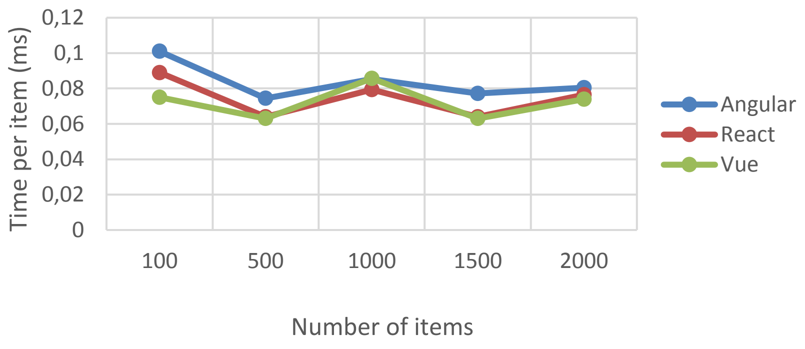

Fig. 4. Dependence of creation time per item on number of items

As shown in Figure. 4, we can see that Vue is the fastest regarding the creating feature. Anyway, because the difference is very small, and probably it can differ regarding the number of rows, we can agree that they all have the same creating time.

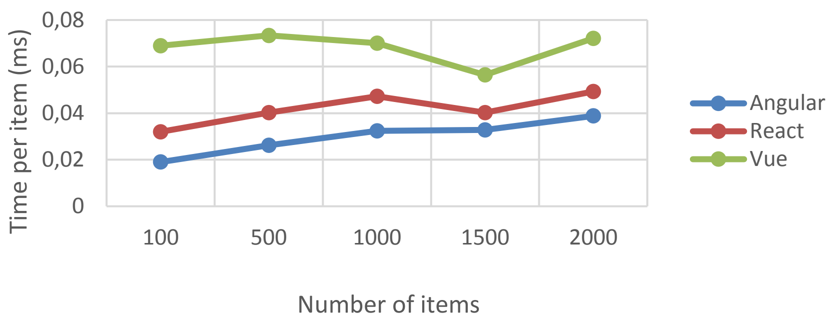

Fig. 5. Dependence of editing time per item on number of items

Figure. 5 shows that Angular is the fastest regarding the editing feature, while Vue has the longest modifying time. We can also see that as the number of items increase, so does the time per editing of a single item.

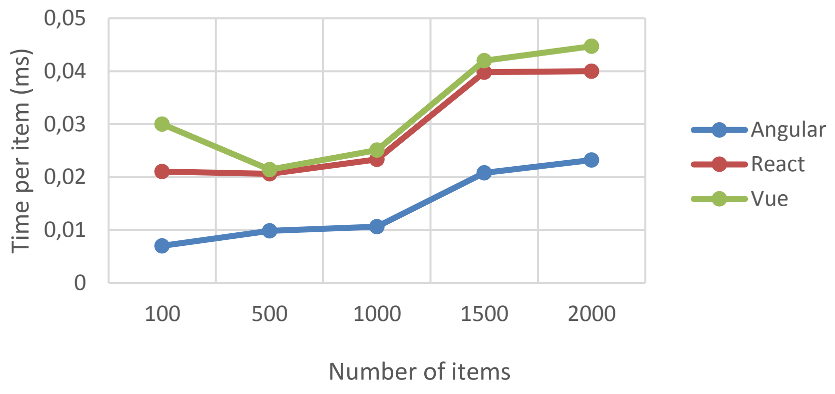

Fig. 6. Dependence of deletion time per item on number of items

Figure. 6 shows that Angular is the fastest regarding the deleting feature, while React and Vue perform much worse and do not differ significantly.

These results make it clear why Angular has the best point score for DOM-Manipulation in Table 1. React and Vue use virtual DOM, but every time a state changes, it needs to be rendered again. However, Angular does change detection against the model earlier and does not require an additional step for rendering virtual DOM.

Memory Allocation

We test the memory allocation by filling arrays with data and then moving the data to a new array. Google Chrome's runtime profiler is also used. The test was repeated a few times on different array size.

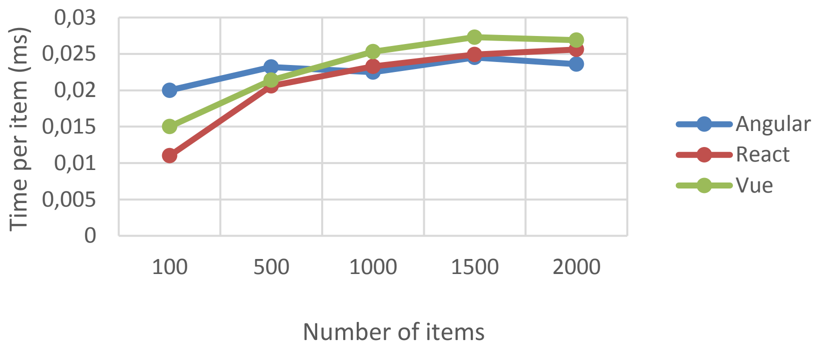

Fig. 7. Dependence of relocation time per item on number of items

On smaller size arrays, React and Vue have better times, due to a well-built structure, while Angular is the fastest as much as the size of the array grows (Fig. 7). Regarding the point score (Table 1.), we can see that all of them perform well in separate fields.

Build Size

The smaller the size of the application is, the faster it is expected that it will run. Logically, this will be the case. But the tests showed us that an application can run fast enough, despite the big size before build. Even if Angular had the biggest size before the build, after the build, the size of all of them is almost identical (Fig. 8).

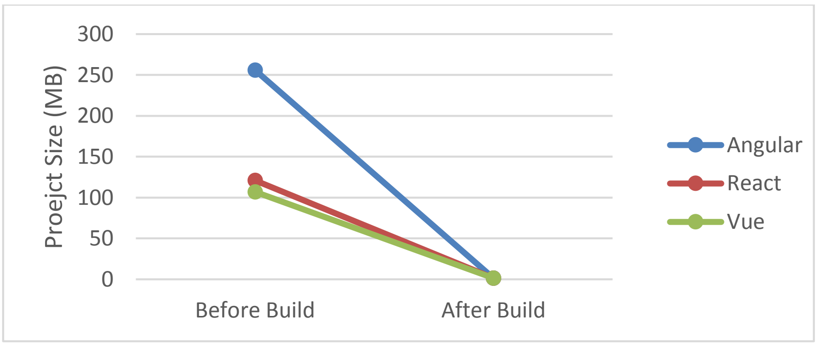

Fig. 8. Dependence of the project size – before and after build

Startup Time

The startup time is the time needed for the page to be fully loaded and ready to use. It is similar to the Speed Index that we observed with the Lighthouse tool. We can get this info on Google Chrome's DevTools, under Network Analysis. In order to get more realistic results, we disabled caching content for this test. As we have seen in the previous test, React has the slowest build size, and yet has the best startup time. Angular and Vue didn't differ a lot, so we give them the same number of points.

Modularity

Rich Package Ecosystem

We can conclude from a comparison of the number of NPM packages available for our frameworks that each of them had packages for every possible web application functionality. If we take that as a point, they should all get the same score regarding this feature. Anyway, React has three times more available NPM packages, so we can say that everything one needs – React has it.

Flexibility

Flexibility can be defined as the possibility to add new functionalities to an already created application. This was made by following the documentation for the frameworks in order to create the application. React and Vue are flexible technologies since it is simple to add their features to an existing project and scale them up later if needed. On the other hand, Angular is not very supportive of flexibility.

Reusability

Reusability is a very important metric of the frameworks for front-end web developments since we are using component-based frameworks. The best way is to create as many as possible independent components, so they can be reused over the project, which is achievable with all the frameworks we tested. In the end, it all depends how the developer will structure his application.

Documentation

The usefulness of the documentation and its ease of use is of crucial importance for a framework, both for starting basic projects and developing advanced features. As they are a long time on the market, the three of them have excellent documentation, deeply explained and with a lot of examples. One thing worth to be mentioned is that code can be tried on the same page while reading the documentation on Vue. That would save a lot of time and you always know if you are on the right track. Another feature of React's documentation that makes it stand out from the others is that it is available in many languages making it accessible to almost every user all around the world.

Learning Curve

The learning curve analyzes If the structure of the framework was like any other, so it would be easy for new developers to learn it.

Angular uses a template-syntax, having additional functionalities which makes the code more compact. It also uses TypeScript, which for many developers, it's a better language than plain JavaScript, but it can be strange and difficult to adapt if the developer is used to JavaScript. Of these three, Angular has the most complex structure and it is harder to be learned [14].

React uses JSX syntax, which is not as readable as the other two. React handles components usually by dividing them into separate folders [15].

Vue also uses template-syntax, making the code more compact. The additional features that come with it, give us the opportunity to remove the unnecessary code and increase the readability. Every component has its own file, which gives a better structure [16].

We can say that developers have a pretty bad experience with Angular, feel comfortable with React and find Vue easy to learn and handle.

When building a simple single-page front-end application, React received the greatest average score in both the non-weighted and weighted results, as shown by the compiled result (Table 1.).

As a support to our research, we compared Angular, React and Vue using Google trends, and, we can see that React is dominating the market now, even though, looking at the past few years, we shouldn't underestimate Vue (Fig. 9) [8][13][17][18].

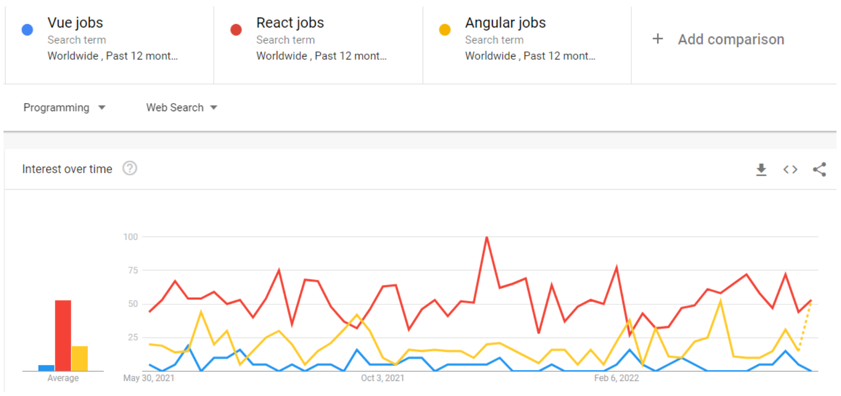

Fig. 9. Google trends' ranking during the past 12 months

5 Conclusion

Developers frequently concentrate on qualities like popularity, documentation, talent availability, or community when selecting a framework to adopt in their projects. Even if these factors are significant, we are aware that in the case of modern front-ends, the user experience is the key consideration. And a critical part of it is website speed [13][18].

In our paper, we had 3 main measures, comparing a single-page application created with Angular, React.js and Vue.js, which were crucial to getting the answer to our questions at the end. While using the Lighthouse tool, Angular proved to be the slowest, while React and Vue had impressive scores. Talking about performance, React was considered best, mostly because of the good startup time, which is important in this case. Angular and Vue were not bad in the performance area too. Regarding modularity, React once again got the first place, having a very widespread NPM packages system, which made the difference between React and Vue. Angular was stuck in the last place. And lastly, usability, where Vue took the first place, having very good documentation and being easy to learn.

From our analysis we concluded that React tends to be the most suitable option for a simple single-page front-end application in our case, Vue comes second being the most friendly for the developers, while Angular is in the third place, having the best score only in the DOM-Manipulation tests. React's popularity is also shown on the Google trends chart (Fig 9.), which proves the comfortability it provides to the developers.

This paper can be a helpful guide to every developer, novice, or expert, aiming to start a new project, or recreate an old one, giving him the right recommendation depending on the type of the application. It is important to understand that it gives a better user experience using the right technology, even when the application is simple, having a good performance application is crucial.

As future work, in order to get a better picture of the front-end web applications, our analysis can be extended to other frameworks on the market that may be not as popular as the considered ones but may compete or outperform them in certain features. Furthermore, we could also create much complex applications and try to apply other tests criteria such as server performance and use different methods for the analysis. If possible, we can try to answer questions regarding the reasons why a certain framework performed best in a concrete field and get the idea what happens behind the scenes.

References

1. M. Kaluža, K. Troskot, B. Vukelić: Comparison of Front-End Frameworks for Web Applications, Zbornik Veleučilišta u Rijeci, Vol. 6 (2018), No. 1, pp. 261-282

2. Jacek Schae, "A RealWorld Comparison of Front-End Frameworks 2020", https://medium.com/dailyjs/a-realworld-comparison-of-front-end-frameworks-2020-4e50655fe4c1, last accessed on 30/08/2022

3. Hamed Taherdoost, "Decision Making Using the Analytic Hierarchy Process (AHP); A Step by Step Approach" https://hal.archives-ouvertes.fr/hal-02557320/document#:~:text=In%20the%20next%20step%2C%20in,be%20extracted%20from%20Table%203, last accessed on 30/08/2022

4. Andreas B. Gizas, Sotiris P. Christodoulou, Theodore S. Papatheodorou, "Comparative Evaluation of JavaScript Frameworks", https://www.researchgate.net/publication/254008963_Comparative_evaluation_of_JavaScript_frameworks, last accessed on 30/08/2022

5. YongKang Xing, JiaPeng Huang, YongYao Lai, "Research and Analysis of the Front-end Frameworks and Libraries in E-Business Development", https://www.researchgate.net/publication/332456776_Research_and_Analysis_of_the_Front-end_Frameworks_and_Libraries_in_E-Business_Development, last accessed on 30/08/2022

6. V. Gonzalez-Prida, L. Barbera, P. Viveros, A.Crespo "Dynamic Analytic Hierarchy Process: AHP method adapted to a changing environment"

https://reader.elsevier.com/reader/sd/pii/S1474667015338908, last accessed on 30/08/2022

7. Piero Borrelli, “Angular vs. Vue vs. React: Comparing frameworks by performance”, https://blog.logrocket.com/angular-vs-react-vs-vue-a-performance-comparison/, last accessed on 30/08/2022

8. Codeinwp, “Angular vs React vs Vue: Which Framework to Choose”, https://www.codeinwp.com/blog/angular-vs-vue-vs-react/, last accessed on 30/08/2022

9. Layer0, “Optimizing React, Angular or Vue Single-Page Applications for Performance”, https://www.layer0.co/post/optimizing-react-angular-vue-single-page-applications-performance, last accessed on 30/08/2022

10. F. Monteiro, Learning Single-page Web Application Development, 1st ed. Packt Publishing, 2014.

11. N. Ockelberg, N. Olsson, Performance, Modularity and Usability, a Comparison of JavaScript Frameworks, https://www.diva-portal.org/smash/get/diva2:1424374/FULLTEXT01.pdf, last accessed on 30/08/2022

12. Layer0, “Optimizing Your Website for Lighthouse v6.0”, https://www.layer0.co/post/lighthouse6-website-frontend-optimization, last accessed on 30/08/2022

13. Tech Magic, “React vs Angular vs Vue.js — What Is the Best Choice in 2022?”, https://www.techmagic.co/blog/reactjs-vs-angular-vs-vuejs-what-to-choose-in-2020/, last accessed on 30/08/2022

14. Angular, official documentation, https://angular.io/docs, last accessed on 30/08/2022

15. React, official documentation, https://reactjs.org/, last accessed on 30/08/2022

16. Vue, official documentation, https://vuejs.org/guide/introduction.html, last accessed on 30/08/2022

17. Star History, https://star-history.com/#facebook/react&vuejs/vue&angular/angular, last accessed on 30/06/2022

18. Google trends, https://trends.google.com/trends/explore?cat=31&date=today%2012-m,today%2012-m,today%2012-m&geo=,,&q=Vue%20jobs,React%20jobs,Angular%20jobs, last accessed on 30/06/2022