A Study On Evaluation Of Browser Performances Of Single Page Application Frameworks

Dissertation submitted in part fulfillment of the requirements
for the degree of
Master of Science in Data Analytics
at Dublin Business School

Mr. Saurabh B. Devade
(10531140)

MSC In Information Systems With Computing 2019-2020

DECLARATION

I, **Saurabh B Devade**, declare that this research is my original work and that it has never been presented to any institution or university for the award of Degree or Diploma. Also, I have referenced correctly all literature and sources used in this work and this work is fully compliant with the Dublin Business School's academic honesty policy.

SIGNATURE
Saurabh B. Devade
DATE: 25/08/2020

ACKNOWLEDGEMENT

Firstly, I would like to thank and express my sincere gratitude to my thesis guide and teacher Dr. Shazia Afzal. Prof. Shazia Afzal has been very helpful whenever I approached her for any doubts or guidance. She made sure that my work remains unique and will support me in my future endeavours also. She reviewed the report several times and pointed out the necessary changes to be made. She was always available for any discussion over the mail or online meetings. She was also my professor In my postgraduate studies for the subject of Data Analytics & Advanced Databases and she was very helpful then also.

I would also like to thank the Dublin Business school for allowing me to pursue my postgraduation study and allowing me to do this work.

I also thank my parents, sisters & friends who encouraged & supported me throughout my studies to do well.

ABSTRACT

Every day new websites are getting launched for many reasons. People are choosing online platforms to support their business or they are building their businesses entirely online. Also, the internet is becoming cheaper and easily available these days. Not just this but technology is growing so rapidly that everyone has access to smartphones and the internet. Due to all these reasons, the importance of the website has increased tremendously. Also, the approach to developing websites has changed from multi-page websites to single-page applications for better user experience, reliability, performance & rapid development. Single Page Application is better in terms of speed and it saves a lot of resources, therefore SPA has become most developer's first choice when it comes to web development. But there are a lot of frameworks has evolved in recent year which supports the development of SPAs. So there is always ambiguity of which framework to choose depending on their need and final goal. While selecting the right framework is very essential and this is the primary phase of designing the application. The performance of the framework in the client's browser plays a very important role while choosing the framework. Therefore this study tries to evaluate the browser performance of four major JavaScript frameworks used for building SPAs.For this study framework chosen is Angular 8, React, Vue.js & Ember.js. The frameworks will be tested for the metrics FCP, SI, LCP, TTI, TBT & Time taken to render the n number of lists. To test these frameworks a Netflix movie app is created in all the four frameworks and tested in the browser. And their results are discussed.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
CLI	Command Line Interface
CSS	Cascading Style Sheet
DOM	Document Object Model

FCP	First Contentful Paint
HTML	HyperText Markup Language
IDE	Integrated Development Environment
JSF	JavaScript Frameworks
JSON	JavaScript Object Notation
LCP	Largest Contentful Paint
LLOC	Logical Lines of Code
LOC	Lines of Code
MVC	Model View Controller
OS	Operating System
REST	Representational state transfer
SEO	Search Engine Optimization
SI	Speed Index
SLOC	Structural Lines of Code
SPA	Single Page Application
SQL	Structured Query Language
TBT	Total Blocking Time
TTI	Time To Interactive
UI	User Interface
URL	Uniform Resource Locator

UX	User experience
XML	Extensible Markup Language

CHAPTER 1: INTRODUCTION

Undoubtedly the use of the internet is growing day by day because it is available for cheaper these days and easily available. With the increase in the use of the internet the importance and use of websites have also increased. *(Khurana and Kumar, 2019)* Smallest to smallest information is also available on the internet and websites are the only sources of that information. Organizations and companies are hugely relied on the web to deliver their products and services to the customers and they have succeeded. *(Lawrence, 2017)* Not just this but the technologies available for building the websites are also evolving. The technologies used for building websites include HTML, JavaScript, CSS, etc. But to fulfill the current market needs the traditional approach to build websites are not considered good enough in terms of its efficiency, performance, etc*(Mesbah and Deursen, 2007; Wang et al., 2008)*. So to overcome this the new approach of the web has evolved. This approach is called Single Page Applications*(Mesbah and Deursen, 2007; Fowler, Denuzière, and Granicz, 2015; Molin, 2016)*. Using this approach building websites has become much easier not just that but websites have become more responsive and efficient and resource savings.

1.1 Conventional Approach

As discussed above the websites built in older times were not so advanced. Most of the websites were built using server-side rendering technology. In the server-side rendering, all the data that needs to be shown on the browser was already fed to the HTML page and that page is sent to the client-side. So all the content will be visible in one go at the browser. So if the size of the webpage is large it would take significant time on the browser to display the content. Also if one has to update only the specific part on the webpage, the whole page has to reload, which means all the data will be fetched again from the server and displayed in the browser. But it has a major disadvantage that even the static data will also be downloaded again with the

updated data and that is wastage of resources. Static data like headers, footers logo, etc doesn't need to be downloaded again. This increases network latency and interfaces complexity. This pattern is also called the request-wait pattern. *(Mesbah and Deursen, 2007; Wang et al., 2008)*

Also, while writing the code, code repetition should be avoided as much as possible. But in the older approach, the content which should be displayed on all the pages has to be written separately which is bad practice. *(Moreno and Robles, 2015)* To overcome issues of Reloading the entire page on every interaction, wastage of resources, increased loading time, responsiveness, catching capabilities, debugging in the browser, etc. new technologies were introduced called a single page application SPA.

1.2 Single Page Applications & AJAX

Single-page applications are composed of several different independent components that can be updated or replaced dynamically so that the enter web page is not required to reload on every interaction of the user with the webpage. Single page application runs inside the browser instead of fetching data every time from the server. Because of this, the websites have become more interactive and dynamics and innovations are being added day by day. Also these days the user-friendly ness of websites cannot be ignored. *(Davila and Navon, 2015; Voutilainen, 2017)*

Single-page applications are heavily dependent on the AJAX. AJAX is Asynchronous JavaScript And XML. As the core technology of the web, AJAX has got more and more attention. Because as mentioned earlier the quality of user interactivity cannot be ignored in modern-day websites so the rich client-side technology called AJAX evolved. AJAX is nothing new but just a new way of using the old standards. All the work of fetching data from the server in runtime and reducing the server side and client side time is done by the AJAX and it helps to improve the user experience. Traditionally XML was the choice of data format for the data

to be transferred but Nowadays JSON is chosen. *(Smith, no date; Lin et al., 2012; Jadhav et al., 2015)*

1.3 JavaScript Frameworks

The existence of JavaScript frameworks is since 1995, and it is one of the most widely known and used programming language for front-end web development. Still, JavaScript is growing and many new features are available now to enhance the user experience. With the evolution of JavaScript language for web development, many JavaScript frameworks are also coming out to support web development. Frameworks are nothing but a set predefined code and functionality which user can use and enhance them according to their needs. Writing everything from a scratch is time-consuming therefore the use of frameworks comes in handy. Also, managing and maintaining code is as important as developing it. So architecture provided by frameworks is also important in this place. A typical JSF will provide functionality like DOM traversal and manipulation, AJAX manipulation managing layout and inserting effects, MVC architecture, URL redirecting, Routing, Stage management, Session Management, etc.

1.4 Rationale

As discussed above to improve the web development to the next level there is a jungle of JavaScript frameworks are available. Sometimes developers may choose more than one framework to fulfill the requirement of applications, so there is always a conflict of which framework to choose. *(Graziotin and Abrahamsson, 2013)* Also, it's not just about choosing the right framework but there are so many performance factors to be considered while choosing the framework. One of the important aspects of these frameworks is its performance in the browser. There are many more factors that may be considered while choosing the frameworks it includes factors like maintainability, validity, Lines of code, Active community. Therefore this research work will be done to compare the latest Single Page Application frameworks and how they perform in the browser. So based on performance and need developers can choose

the framework. This work will mostly try to evaluate the browser-specific performance of the number of different JavaScript frameworks. The chosen JavaScript frameworks and performance parameters will be discussed in-depth in the next chapters.

1.5 Research Question

This research will mostly try to evaluate the browser-specific performance of JavaScript frameworks in the browser. Because if your browser is taking more than 3 seconds to show a particular website then there are high chances of users navigating to some other website. So the website's speed index is one of the important factors. So based on the above criteria following research questions have been formed.

1. How to evaluate the browser performances of the JavaScript frameworks for Single Page Applications (SPA)?

1.6 My Interest In The Topic

Having worked as a full-stack developer for more than two years, I realized the importance of frameworks and how they can play a major role in developing any kind of software. Not just in JavaScript but any other language like Java, PHP, Python, etc.

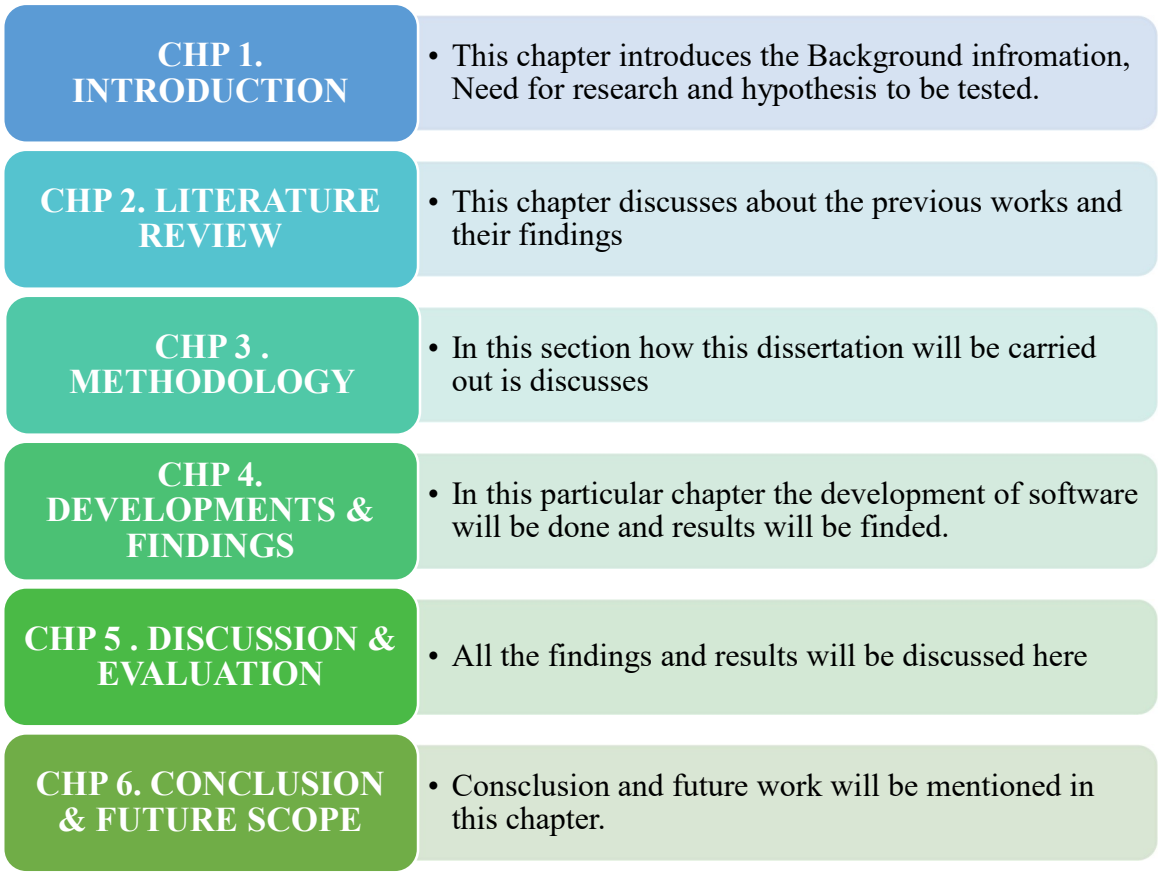
So the main hypothesis to be tested is, using various JavaScript frameworks can we build a robust, scalable single-page web application. Also how the performance of this application differs in the browser in the same environment. The amount of code required to bootstrap the application so it can be maintained in the future will be tested.

The reason for choosing the above topic is because I am very much interested in software development and I would like to pursue this field as my career. Therefore, I wanted to test the different frameworks as I have some experience working with them. Also in my taught masters, I had the subject related to the Front-End web development and Android development where I have worked with the JavaScript, JQuery, Angular 8, and Ionic framework using Angular 8 and hence I wanted to focus on this specific subject area.

1.7 Roadmap for the dissertation

This dissertation is carried out in the following steps.

- Chapter 2 the details about the previous research have been discussed. Their methodologies, frameworks used by them, and metrics used by them are discussed.
- The next chapter is the methodology, in this chapter how this research will be carried out, which frameworks will be chosen what metrics will be chosen is discussed.
- The next chapter will be the development and results. Here the development of the application using different frameworks will be discussed and after the development, their performance will be computed. After the development,
- the next chapter is evaluation and discussion where all the findings and results will be discussed.
- And the last chapter will be the conclusion and future scope.



Conclusion

This chapter covers the information about how websites were used to build using a conventional multi-page approach. And how it got transferred to single page application over

the period. Then it talks about what is the JavaScript frameworks, their importance. Then in the next section, it talks about why I chose this topic for research and my interest in this field. Also, the Research question is mentioned. And finally, this chapter concludes with how this entire thesis will be carried out

CHAPTER 2: LITERATURE REVIEW & RELATED WORK

This work is all about comparing the different latest JavaScript frameworks and evaluating them, which is nothing but benchmarking. So this chapter will discuss the background information on what is benchmarking and its importance in the field of technology, What are the frameworks and their importance

2.1 Benchmarking

According to the various sources, there are different ways of defining the benchmarks but more or less what is mean is the same. Here are some definitions of benchmarking by various sources, the International Organization for Standardization and the International Electrotechnical Commission has defined benchmark as *"A standard against which results can be measured or assessed"* in the same way IEEE defines benchmarking as *"A standard against which measurements or comparison can be made"* According to to the Bouckaert, Philips & Wallander the computer benchmarking can be defined as computer benchmarking is "*The act of measuring and evaluating computational performance, networking, protocols, device, and networks, under reference conditions, relative to a reference evaluation*" *(Bouckaert et al., 2011).*

Now that definition of benchmarking is clear, let's see what are the computer benchmarking and their types and understand their importance.

Benchmarking is not a new concept. Benchmarking has been around for more than decades. It's being used for comparing various platforms, different tools, different technologies, and their performances. It finds out differences between different components of the same families and tries to evaluate them. *(Lawrence, 2017).* Benchmarking tools are nothing but the solutions which are developed to automate the process to evaluate the different metrics of different application in the customer environment. Benchmarking comes in handy for the organizations or individuals while choosing a particular tool or technology for their own needs, so while choosing that they can compare the different aspects of solutions instead of choosing one solution every time thinking of it as an only correct solution. Benchmarking can also be helpful for the ones who are developing a new solution, so they can compare their result with the old solutions and get an idea of their solution to improving so overall it will help in standardization. *(Ratanaworabhan, Livshits and Zorn, 2010).*

There are two major categories of benchmarks i.e micro-benchmarks & macro benchmark. Using micro-benchmarks a very small part of the portion of the application is evaluated or analyzed on the other hand in the macro benchmark are designed to analyze the large and complex system on a bigger scale. The type of benchmarking used for this work will the microbenchmark because we will be evaluating a few parameters of an entire web application. *(Seltzer, Krinsky, Smith, Xiaolan Zhang, Harvard Uni).* Now that we have a sufficient understanding of what benchmarking is and what is its importance. In the next section, let's understand what are JavaScript frameworks and its importance.

2.2 Frameworks & JavaScript Frameworks

In the introduction section, the concept of frameworks and JavaScript frameworks is discussed briefly. But in this section, we will try to get into a deeper level and try to understand what are they and their importance.

2.2.1 Frameworks

Frameworks can be defined in many different ways. According to Ralph E. Johnson, *"Framework is nothing but a reusable system build to use in all part of software which is represented by predefined abstract classes and the way their instances interact"(Johnson, 1997).* Further, he gives the second definition as *"Framework is a skeleton of software which can be modified the way developers want to for satisfying the needs according to the software" (Johnson, 1997).* These two definitions are not different, there phrasing is different but what they mean is the same. The first definitions talk about the way it works while the second one talks about its structural aspect. Moreover, both definitions simplify the difficulty of defining frameworks. *(Johnson, 1997).* Therefore it can be said that frameworks are build to allow developers to solve the problems that frameworks are capable of. *(Schmidt and Buschmann, 2003).*

We can all agree that how technologies have become competitive & challenging over the years. Therefore, there are certain characteristics that any tool or frameworks should possess to sustain in this highly competitive market. These characteristics are as follows

1. **Affordability:** Affordability states that the total ownership costs of software acquisition and should not be very high.
2. **Extensibility:** It should be compatible with the new updated to address new requirements so it will be adopted in the market even for new requirements.
3. **Flexibility:** Should support the emerging technologies
4. **Portability:** It should not be bounded to a specific environment like OS. It should be multipurpose and platform or environment independent.
5. **Reliability:** To make sure that built applications are robust and tolerant of future faults.
6. **Scalability:** The application should be able to handle a large number of requests or clients simultaneously.
7. **Trustworthiness:** To ensure integrity confidentiality and availability in distributes systems. *(Schmidt and Buschmann, 2003)*

2.2.2 JavaScript, JavaScript Frameworks & SPA frameworks

JavaScript was designed by Brendan Eich at Netscape. It is an object-oriented programming language focused on nonprogrammers to extend the support for client-side code execution. It

does not concept of classes and does not support encapsulation. It does not even have structured programming like other programming languages like JAVA, C, C++. JavaScript believes in flexibility. No one can deny the success of JavaScript, if we talk about the numbers 97 out of 100 websites use JavaScript as their client-side scripting. Sometimes this language is also referred to as general-purpose programming language. *(Richards et al., 2010).* Initially, it was named LiveScript but later it got renamed to JavaScript fun sun and Netscape started shipping JavaScript with the Netscape browser in Dec 1995. JAVA was the only language that used to run in the browser, but it was very heavy to execute, so JavaScript came as an alternative. Also, it was targeted for the less experienced developers. *(Voutilainen, 2017).* JavaScript may look like other languages by syntactically for eg C, C++, or JAVA but it is a loosely typed language. For example, while defining any variable in JavaScript you don't have to type the data type like Integer or String you can just type 'var' and there you go, your variable is defined.

The popularity of JavaScript has grown over the years with the development of the web. The scripting of webpages has become more complex due to the evolution of a technology called AJAX as the webpages have become more complex, unlike static pages in older times. Web giants such as Amazon, Facebook, Gmail contains a significant amount of JavaScript code. Web Apps have become more popular because they don't need any additional information software mechanism they are OS independent i.e they can work on any platform like Windows, Android, Linux all they need is one browser with an active internet connection. *(Ratanaworabhan, Livshits and Zorn, 2010).*

When AJAX evolved and with the help of jQuery it became easy to update certain parts of the webpage dynamically. Because of jQuery, it became easy to manipulate the DOM and update the data fetched from the server. These pages were interactive but it is nothing like a single page application that exists today. In jQuery, one has to find every element on the DOM to

manipulate it with the help of either CSS class or with the element id or with the element name. It was not that efficient. *(Voutilainen, 2017)*

Therefore, the concept of client-side JavaScript frameworks became popular around 2009. JSF helps in binding the HTML page with the JavaScript code with the help of data binding so we don't have to do this explicitly as we do in jQuery. Also, JSF's make the processing of fetching data from the server and updating it in DOM a lot easier. It takes care of all the routing needed in the application. Also, it helps in managing the code structure, plus the separation of codes this all makes the frameworks very important. These frameworks are called SPA frameworks. The definition of SPA is already mentioned in the introduction section here we will see what are the basic attributes of the SPA's.

1. **Components**: The page to be displayed is divided into different parts while developing called components, but while rendering it shows all as one page.
2. **Web interface**: The interaction between a user and a web server.
3. **Update**: It is possible to update, delete, or replace one component with another component dynamically.
4. **User action**: User can interact with the page by doing any action with the use of an input-output device, for some action to take place.

2.3 Previous work

Few attempts have been made to evaluate the single page application frameworks in the past. Every work has different ways and metrics to be evaluated. The choice of framework is also different from work to work. This work is the progression of past works.

One of the most recent studies done by *(Lawrence, 2017)* for evaluation of the single page application frameworks. For his work, he has chosen a reference TODO MVC application developed and maintained by Addy Osmani & Sindre Sorhus called TodoMvc*(TodoMVC, no date; Mardan, 2014; Bainczyk et al., 2017)*. This application has been developed in every latest JavaScript from that time (2017). According to him, any other application could be used but

that particular application is maintained by the expert developers so that the app has been chosen. Also, different applications may yield different results. Todo application has an **input field** to add the items in the list, there is also functionality to mark items as completed or uncompleted. Then there is a **task list** which is shown on the webpage each item is editable as well. It also has the **footer section** where items can be sorted based on their status like completed, uncompleted, or all, etc. Frameworks selected by him are BackboneJS, ReactJS & AngularJS.

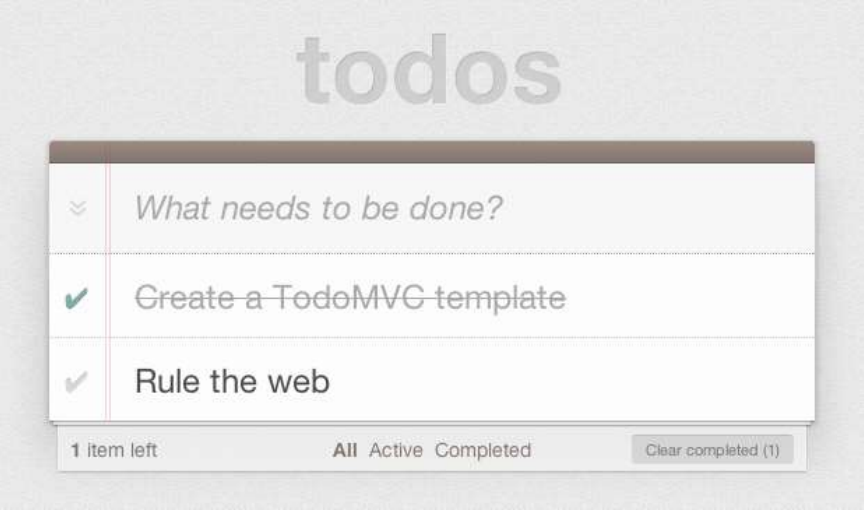


Figure 1 - TODO MVC APP

To perform the benchmarking test there is 3 task to be executed. In the first task, 100 items will be added to the todo list, then the second task is to mark all added task as completed task in which all the 100 tasks added in the previous step will be marked as completed and in the final task, all the completed task will be removed. After the completion of all the tasks, the benchmark application will show the visual report of the performances of all the selected frameworks in this case BackboneJS, ReactJS & AngularJS.

The figure shown below depicts the performance results generated by the Todo MVC application.

Figure 2 - TODO MVC APP BENCHMARK RESULTS

All the tests were executed 25 times in his work to ensure the validity of work in three different browsers i.e Google Chrome, Mozilla Firefox & Microsoft Edge.

After running tests in the chrome browser, the BackboneJS showed the least amount of execution time of 157 ms while React-JSX took the maximum time for execution i.e 904 ms. Angularjs and React-no-JSX take 346 ms and 554 ms respectively.

Moreover, in the Microsoft Edge browser, we can see the significant rise in time take for execution for all the frameworks. But here as well BackboneJS outperformed any other framework. BackboneJS took 233 ms and React-JSX took 1953 ms. much higher

And the same for the Mozilla browser.BackboneJS wins here as well with 266 ms. So Overall he concluded that BackboneJS is the most efficient framework in terms of the execution while React-JSX takes much more time for task execution comparatively. Also, all the frameworks have more execution time in the Microsoft Edge browser compared to the other browser. *(Lawrence, 2017)* .

A very similar approach was used by *(Davila and Navon, 2015)*. This work also used the same TodoMvc application by Addy Osmani & Sindre Sorhus. But the selection of frameworks was different. Frameworks chosen in this work are Angular, Backbone, Ember, Marionette and

React. And very similar tasks mentioned in previous works were performed and results were obtained and evaluated.

According to *(Gizas, Christodoulou, and Papatheodorou, 2012)* quality of framework is one of the most important factors when it comes to benchmarking. Therefore in his work size, complexity, and maintainability of the framework have given more importance. The parameters measures are **Size Metrics**: lines of code(LOC), the number of statements and number of comment lines, and ration between the lines of code and comment lines. The next metric is **Cyclometric complexity**: Which is McCabe's cyclomatic complexity, branches, and depths. In **Maintainability metric:** Halstead metric and maintainability index is measured.

From his result, he found out that YUI3 3.4.1 frameworks take the most number of lines to build the application with the most number of comment lines i.e 12210 & 9624 respectively. Where jquery takes the moderate amount of lines 7252. If we talk about the Cyclomatic complexity in his work YUI3 3.4.1 framework has more number of functions i.e 28 whose complexity is greater than 20 & DOJO 1.7.2 being the framework with the least number of functions whose complexity is greater than 20. Also, several performance tests were run on the five different browsers including Chrome, Safari, Opera, IE9, and firefox. In those tests, he has mentioned some issues revealed by the performance test that is as follows. IE8 has shown the very big execution time for the frameworks MooTools, YUI2, and YUI3.

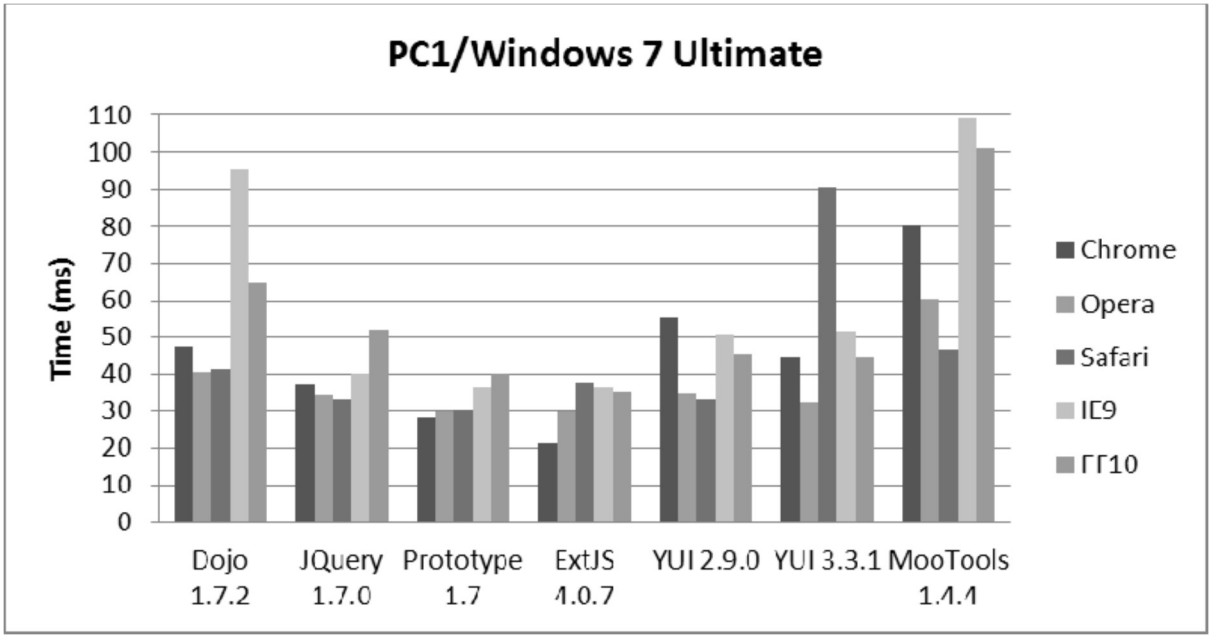


Figure 3 - TODO MVC app benchmarking result

Conclusion

This chapter gives an overview of what is benchmarking & its importance, Also it explains What is single-page applications in depth. Then, At last, the previous attempts to evaluate JavaScript, their approach, and their findings have been discussed.

CHAPTER 3: METHODOLOGY

This chapter provides an overview of the research methodology and methods used to carry out this artifact. This section describes the selection of frameworks, selection of parameters, an application designed for computing the performances, different browsers used, tools and techniques used for obtaining the various results.

3.1 Application Design & System Architecture

3.1.1 System Architecture

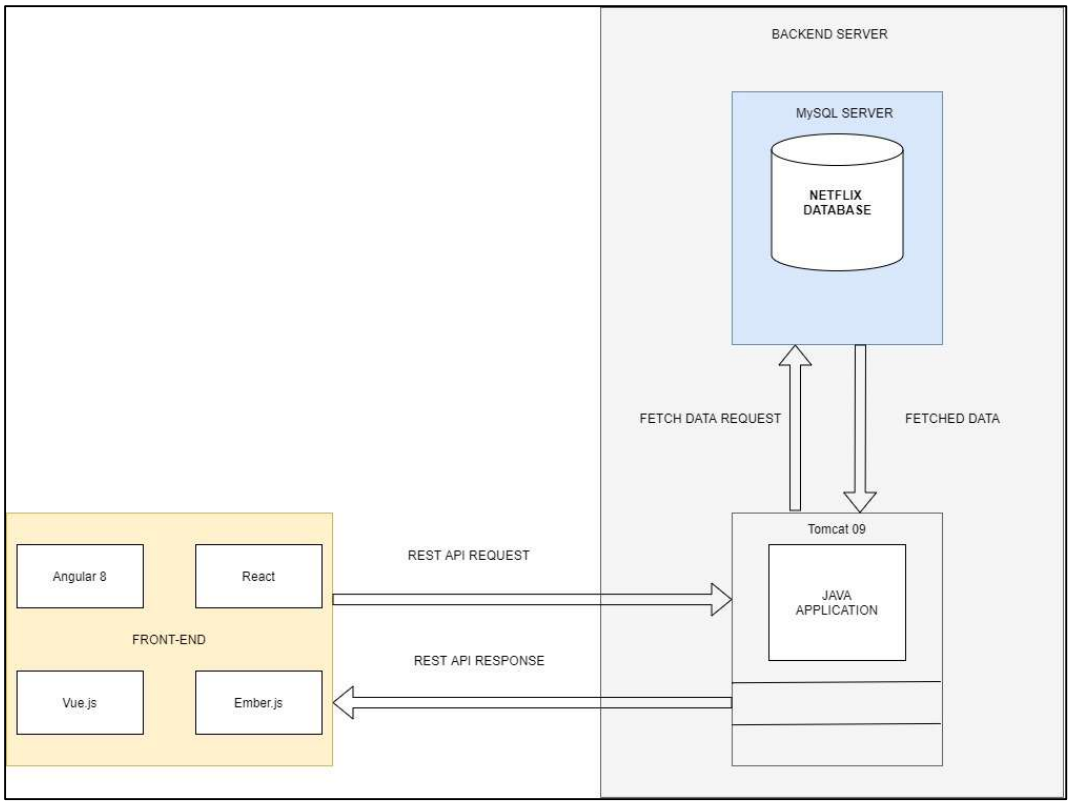


Figure 4 - System Architecture

The above figure shows the entire architecture of our application including back-end & front-end. This architecture is primarily divided into the two portions i.e back-end & front-end. But for this major focus is given to the front-end part of the application which is discussed in more

depth in the coming chapters. So the system works in the following way. Whenever there is a need for fetching data from the server, the front-end application sends the AJAX request to the server using the REST APIs. On the server the java application is running in the tomcat server, tomcat redirects that request to the java app. Once the application receives the request it invokes the method bonded to that API URL. That method executes the logic written inside it and performs the curd operation on the database using the MySQL server. Once the operation is done in the JAVA application it responds with the resultset and operation status. In the front-end application, that response is received and the changes are reflected in the browser as needed. So in this manner, the entire architecture will keep working.

3.1.2 The Netflix Movie app

This research is about computing the performance of various single page application frameworks. So to compare this framework there is a need for web applications built using these frameworks. As discussed in the literature review researcher *(Lawrence, 2017) & (Davila and Navon, 2015)* have used the TODO MVC app created by Addy Osmani & Sindre Sorhus. This is an open-source project built for computing the performance of different frameworks. At the time of their research, this application was well maintained and updated regularly. So it made sense to use that application to save time. The first approach was to use the same app. But this particular app is not being updated and maintained by its developers. In addition to this, most of the frameworks used in that app have updated versions now. So the first approach is eliminated.

So for this study, a new application will be developed using the choice of frameworks. The frameworks chosen will be discussed in a later section of this chapter.

For this study, The Netflix Movie app will be created. As our aim is not to build the app but to study the performance app one smaller portion of the app will be created. Also, the UI is not

important in this specific condition so, the app with basic UI & UX is developed. Also, the TODO MVC app hasn't implemented the whole UI.

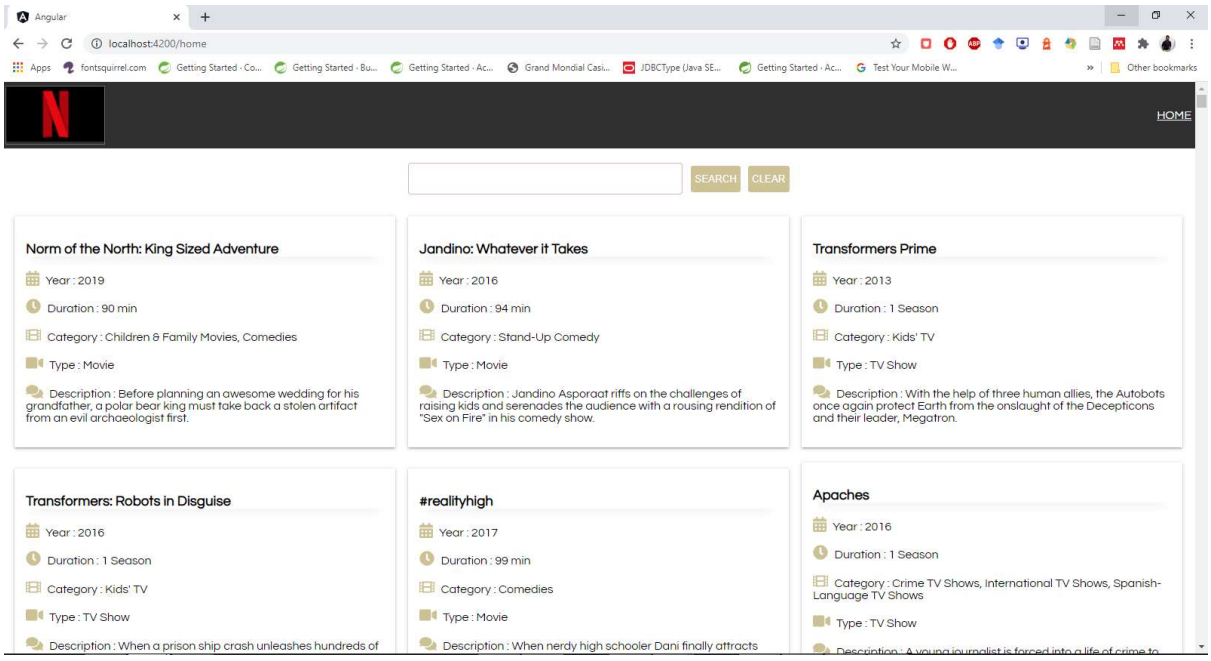


Figure 5 - Netflix App UI

Netflix movie app will have one route "/home" on that route. All the movies and shows of Netflix will be displayed along with their Title, Year Released, Duration, Title, Genre, Rating & with a little description of that movies or shows. The database is taken from Kaggle.com.

Further, this app also will have search functionality. Using this feature visitors can search the movie or shows based on Title, Show cast, Director. Genre, Rating, etc. The clear search feature will also be there.

3.2 Chosen Frameworks

A lot has been said in Chapter 1 & Chapter 2 about the Single Page Application frameworks. This section will talk about the frameworks chosen for this artifact and the reason behind it. Four frameworks are selected for building the Application and they are Angular 8, ReactJS, VueJS & EmberJS. While developing the latest version of all the frameworks are used. While

selecting the framework, for similarity the frameworks with the CLI tool are given preference. CLI tool stands for a command-line interface. It was the measured source of input in all Linux based systems in the 70s–80s. *(Jayakody et al., 2017)* The reason for using command-line tools while developing a web application is to get bootstrap with the application quickly. In very few commands it setups the project directory and it also helps in maintaining the code in the future. And in the next one command application will be running in the browser. It also helps in adding external dependencies to the project easily. *(Farwell, 2017).*

Frameworks	First Release	Latest Stable Release
Angular 8	14 Sept 2016	12 Aug 2020
ReactJS	29 May 2013	19 March 2020
EmberJs	8 December 2011	4 May 2020
Vue.js	February 2014	13 December 2019

Table 1 - Framework release version information

Let's see the frameworks selected in more detail.

3.2.1 Angular 8: Angular is a google's JavaScript framework and it was created by two google developer Misko Hevery and Adam Abrons. When it was created it was based on plain JavaScript and it was named as AngularJS. At that time majority of websites developed were Multi-Page website and the cons of those have been already discussed. jQuery library was famous at that because using AJAX with jQuery is much easier than the plain JavaScript and it was easy to manipulate the DOM with jQuery. But AngularJS went one step ahead of jQuery because of features like two-way data binding, route handling, support for the external library, etc.*(Studiengang Bachelor et al., no date).* The reason behind the creation of it was to extend the HTML capabilities with the help of custom directives made by angular. The next update of the angular framework was different from its last version. Which was called Angular 2? The version used in this framework is Angular 8.0. The major difference was the AngularJS version was based on JavaScript while all Angular 2+ versions are based on the typescript.

The structure of the angular framework is component-based like many other frameworks. Components are the main building blocks of the frameworks. They are used excessively throughout the application. Each component can be associated with the separate HTML sometimes referred to as template & CSS file. All the business logic, data manipulation, arithmetic, and mathematical operations are written inside the components.

Between all the components there is one main component called app-root component which helps in initializing the entire application. The components are hierarchically like parent and child. And it is possible to pass the states between parent and child components.

Data binding is also one of the most important features provided by angular frameworks. Data Binding is the connection between actual data and the front UI of the application. For showing the data fetched from the server or any other source on the UI we have to establish a connection between the template and the components. Angular has different ways of a binding component variable to the template.

Property Binding: In property Binding the direction of the flow of data is from component to template.

Event Binding: Event binding is the opposite of the property binding. Data always flow from the template to the component.

Two Way Data Binding: Two-way data binding is the most important, if not most important then one of the most used data bindings throughout the application. Angular keeps track of every variable using watchers. As soon as the data changes in input filed the values in components also get changes and vice versa.

Routing provided by angular is a mentionable topic. Routing plays an important role in any website. Without proper nesting of routes, the app won't be considered as a good app. While

generating the app using the command, angular ask whether the routing module should be included or not. If selected yes it will add a routing as a separate module and we can add as many as routes we want and we can nest one route inside the another.

3.2.2 ReactJS: ReactJS is maintained and developed by Facebook Inc. It was found in 2003 by Facebook's Jordan Walke as open-source software and since then it has gained a lot of attraction by developers. React is nothing but 'V' in the MVC architecture or It can be said that It is used for just creating user interfaces. *(Lawrence, 2017)*.The primary object of the react is to enhance the performances of the applications. It was mainly focused to address the performance issues of the web applications. The most famous use case of ReactJS is Instagram & Whatsapp. Most often react is mistaken as a framework but it just a library to build a UI component. The difference between library & Framework is that framework help you decide the entire application, your applications are entirely dependent on that single framework it makes developer's job easy while in case of the library, you can just import it and it will you help you build that particular smaller portion of the website. *(Voutilainen, 2017).*

Virtual DOM is the main reason why react is gaining more importance over frameworks like Angular our VUE. *(Javeed, 2019)*. With the help of virtual DOM react decide whether the component should be reloaded or not based on the current state of the component and the changes that have occurred.

One way data flow is used in the ReactJS which helps in controlling the flow of data within the application which improves the stability and to detect the changes occurred.

To re-render the component, state and properties play a major role in deciding that. When there is a transfer of properties or states from parent to child the react DOM compares them with the

previously stored value and if there is a difference between the new value and previous value then only it re-renders the component. *(Javeed, 2019).*

Components are the major core of the ReactJS library like Angular. To transfer certain states of the component to the view and display it is the main purpose. The component can be written in two ways i.e one as a function and one as an ES6 class. *(Le, 2020).*

Some important features of ReactJS

- SEO Friendly: ReactJS pages are SEO friendly since the main aim of ReactJS is a performance improvement, rendering of pages is fast in ReactJS therefore they are SEO friendly. *(John, 2010)*
- Testing: It is easy to do testing with the reacts. *(John, 2010)*
- Code Stability: The data flow is unidirectional which keeps the hierarchy very stable. Even if there are some changes in the child component it does not affect the parent. *(John, 2010)*
- Performance: Because of the virtual DOM and Server Side rendering performance of the react app is very fast compared to the others. *(John, 2010)*

3.2.3 Vue.js: Vue.js was introduced in the market (2014) year later of ReactJS's release. Vue.js came out in February 2014 and it was developed by the Evan You who is the former developer of google where he worked a lot on the AngularJS framework. Vue.js is considered a progressive web application framework and it relies on the principle of the Model-View-ViewModel (MVVM) principle. This framework can be used for smaller as well very high and Single Page Applications as well. One measure feature of Vue.js is its ability to scale the applications.

Vue.js is entirely an open-source project build by the single developer as a hobby initially and not by any large funding company.

Like angular and react Vue.js is also built over the components.

Data Binding: Vue.js gives two options to bind the data between the component and its template. One-way data binding and Two-way data binding. *(Shen, Sun and Li, 2018; Song, Zhang and Xie, 2019)* Most often two-way data-binding will be used to synchronize the changes. i.e whatever the changes are being done on the view will always be in sync with the component and vice versa. But it doesn't happen in the one way data-binding. *(Studiengang Bachelor et al., no date)*

One-way: v-bind

Two-way: v-model

Directives: Similar to angular Vue.js also makes use of directive. Directives in Vue.js are prefixed by v-. Directives can be used for data binding, property binding, event handling, and more.

Examples of Directive :

- <p @click="someFunction"> ... </p>
- <button v-on:click="getData"> ... </button>

3.2.4 Ember.js: Ember.js has released in December 2011 by the Ember core team under the MIT license as the open-source project. The original author of the framework is Yehuda Katz. Like the applications mentioned above, it allows developers to build scalable single-page applications. Many Popular sites like Yahoo, Groupon, Discourse make use of Ember.js. *(Shrestha, 2015)*

A router is the most important and powerful concept of the Ember.js. It emphasizes the importance of the URL in managing the state of the application. According to Tom Dale one of the lead dale of Ember.js, when it comes to the latest web application URLs are not just the

way to uniquely locate the pages on the server but they serve the more purpose to the web application. *(Dale, 2012)*

Model: Every route is associated with the different model which contains the data associated with the state of that route. This model can be updated or retrieved by the server using AJAX technology and based on that view can be updated very easily.

Controllers: A controller acts as a bridge between the model and its associated view. All the business logic like form handling, data manipulation, or mathematical applications can be written inside the controllers.

Now we are aware of the frameworks which will be used in this study, lets see the various parameters which will be computed and evaluated to analyze the performance of these frameworks.

3.3 Performance Parameters & Methods To Compute Those

The main objective of this work is to calculate the speed indexing of the web application in the browser. So in this work, only the parameters which reflect performances in time are given the importance. Also, a few parameters associated with the code are taken into consideration.

3.3.1 Lines of codes: Lines of code is one of the most important and oldest metrics around there. Iw was used for the first time in the year 1960 in economic, social studies & productivity it came out as an effective metric. In the era of assembly languages, this metric was quite simple but as soon as languages starting evaluated and it started moving toward a structural programming approach like C language, this concept started getting complex. Therefore later this metric got standardized and two counting methods *(Rentrop et al., 2006; Lawrence, 2017)*.i.e SLOC & LLOC. SLOC is the physical lines of code whereas LLOC is the logical

lines of code. SLOC gives the measure of the number of physical lines in the code excluding the comments and LLOC gives the measure of the number of executable statements within the code. *(Park, 1992).*

Method to count LOC: We will be using two tools to count the Physical lines and And source code lines. Tools used are sloc and cloc and these tools are available on npm as well as on Github. *(Dodds, 2020; Kohlhase, 2020)* We just need to simply write npm install command and then these tools will get installed on the machine. And after the successful installation of these tools, we can execute the commands to count the LOC metric. The reason for using two different tools is for greater accuracy.

While counting the LOC we will count only the amount of code developers has to write and not the auto-generated code by the CLI.

Command used – sloc src & cloc src where src is the directory from which we want to count the LOC.

```
C:\Local Disk D\STUDY\Thesis\CODE\FRONT-END\thesis-netflix-angular8\angular>sloc src

---------- Result ------------

            Physical :  533
              Source :  389
             Comment :  82
 Single-line comment :  17
       Block comment :  65
               Mixed :  5
 Empty block comment :  0
               Empty :  67
               To Do :  0

Number of files read :  22

---------------------------


C:\Local Disk D\STUDY\Thesis\CODE\FRONT-END\thesis-netflix-angular8\angular>cloc src
      21 text files.
      21 unique files.
       8 files ignored.

github.com/AlDanial/cloc v 1.86  T=0.04 s (490.2 files/s, 12417.7 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
TypeScript                      13             55             74            208
JavaScript                       2              0              0             75
CSS                              3              7              2             63
HTML                             3              4              1             43
-------------------------------------------------------------------------------
SUM:                            21             66             77            389
-------------------------------------------------------------------------------

C:\Local Disk D\STUDY\Thesis\CODE\FRONT-END\thesis-netflix-angular8\angular>
```

Figure 6 - LOC command

While building the website, the speed taken by the webpage matters a lot. Faster pages are more efficient and they provide a much better user experience. According to kissmetrics infographics, one-fourth of the user we move to the other website if the website they are opening takes more than 3 seconds*(sean, 2011)*. In the case of mobile users, this ratio is even higher. The same study states that 73% of users have faced the use of a slower page load. According to Walmart the if the load time increases by even 1s, then the conversion rate is decreased by 2%. Also, the SEO of webpages should be good so they can rank higher according

to the Google ranking algorithm. Now Google has included the page speed as their criteria that means the pages with the higher speed will be ranked higher when someone searches for something on google*(sean, 2011)*. Therefore obtaining the performances of the webpage has become more important. So the next 5 parameters will talk about the factors related to the performance of the webpage. *(Guard, 2020)*

The tools used for computing these parameters are google's lighthouse tool. This open-source tool can be used for monitoring the performances of the web application as well as, the load times, their speed index, etc. Is also gives the score for SEO. *(Google, 2020; Saif, Lung and Matrawy, 2020)*

3.3.2 First Contentful Paint (FCP):

When the user navigates to the particular URL then it takes some time for the browser to fetch data from the server and display it in the browser. FCP measures the amount of time taken by the browser to display the first content of the DOM. It may be anything for example any image, heading, paragraph, etc. FCP score is a comparison of FCP of your website with real-time websites based on data from the HTTP archive. *(Rahman and Ikbal, no date; Budiman et al., 2018; Rome et al., 2019)*

Figure 7 - First Contentful Paint

3.3.3 Speed Index :

Speed index measures the time taken for the browser to visually display the data on the webpage during the page load. In simple words, the speed index is how many milliseconds it takes to show the visible parts of the webpage on the website. *(Gao, Dey and Ahammad, 2017; da Hora et al., 2018; Furtak and Wittie, 2019).* For capturing the speed index lighthouse captures the video of webpage loading and calculates the visual progression. Then lighthouse uses the node.js Speedline index module to generate the report. *(Irish, 2020)*

Figure 8 - Speed Index

3.3.4 Larget Contentful paint (LCP):

One of the main factors while calculating the performances of the browser is LCP. LCP measures the time taken to display the largest element of the webpage. This helps in an approximation of when the main content of the page will be visible to the user. The available older metrics load or DOMContentLoaded are not useful sometimes because they don't correspond to the important content of the webpage. *(Walton, 2019)*

Figure 9 - Largest Contentful Paint

3.3.5 Time To Interactive (TTI):

The webpage is not interactive as soon as it loads the content of the web page. It takes some time for it to become interactive. Interactive means buttons to become clickable or input fields to become typeable etc. This metric is important because content visibility should not be optimized at the cost of TTI. This leads to very poor user experience and there are high chances of users navigating away from such websites. *(Saif, Lung and Matrawy, 2020)*

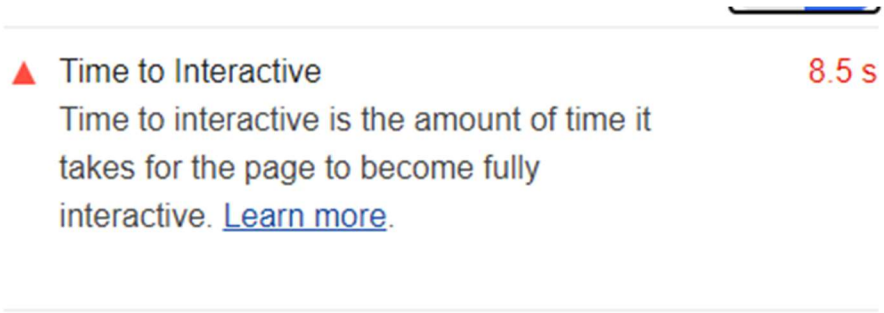


Figure 10 - Time To Interactive

3.3.6 Total Blocking Time (TBT) :

TBT gives the value of time the pages are blocked from responding to any user events. Events like keyboard inputs, or mouse inputs or touch inputs. Lighthouse derives this value from summing all the blocked portion of long tasks between FCP and TTI. Long tasks are very heavy and they keep the main thread very busy at the cost of interactivity and this is not a good practice. Some times long task makes the pages irresponsive as well. *(Osmani, 2019).*

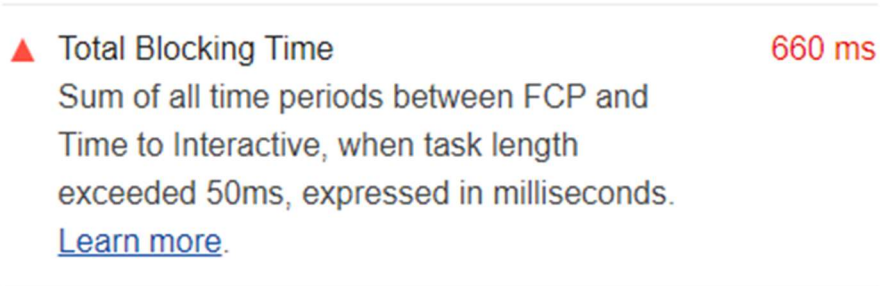


Figure 11 - Total Blocking Time

3.3.7 Time Taken To Render The Real-Time Data:

This study is all about Single Page Applications. In single page application, real-time data i.e data fetched from the server plays a major role. Because in the same URL different data has to keep updating. That's the whole point of the single-page applications. So this custom test is build to test how different frameworks work with the dynamic data. So to test this search functionality has been implemented.

In this test, the user can search the movies of his interest. He can search by title, cast, director, genre, rating, etc. Once the user clicks on the search button this query will be sent to the server using AJAX and based on that search query all the movies related to that search query will be returned from the database.

We make use of **performance.now()** method. This method returns the one **DOMHighResTimeStamp** which is accurate up to 5 microseconds. The returned value represents the time elapsed since the time origin. This functionality is available in web workers.

```
searchSHows() {
  console.log("START", performance.now());
  const jsonBody = JSON.stringify({ title: this.seacrhQuery });
  const config = {
    method: "post",
    url: "http://localhost:8080/netflix/searchSHows",
    headers: {
      "Content-Type": "application/json",
    },
    data: jsonBody,
  };
  axios(config)
    .then((response) => {
      this.shows = response.data.resultSet;
    })
    .catch((err) => {
      // Manage the state of the application if the request
      // has failed
      console.log(err);
    });
},
```

Figure 12 - Time Taken To Render The Real-Time Data

As shown in the above code snippet when the user enters the required query in the input field shown on the webpage and then he clicks on the search button this function gets executed. The first line of this function is **performance.now().** Here the time origin is started and value is logged to the console. This function calls the API written in the back-end. After successfully fetching the data the result set is fed to **this.shows** variable. Now our **this. shows** variable has the array of objects containing all the shows based on the user's search criteria. As soon this variable gets the new value the framework detects the change has happened and it updates the template with this newly received data.

```
<div class="flex-container">
  <div v-for="show in shows" class="show-item" :key="show.netflixShowId">
    <div v-if="show.netflixShowId==6229">
      <p v-bind:style="{ 'background-color': calculateRenderingTime() }">LAST</p>
    </div>
    <div class="item-header-container">
      <h3>{{show.title}}</h3>
    </div>

    <p>
      <font-awesome-icon icon="calendar-alt" class="icons" />
      Year : {{show.yearRelased}}
    </p>
    <p>
      <font-awesome-icon icon="clock" class="icons" />
      Duration : {{show.duration}}
    </p>
    <p>
      <font-awesome-icon icon="film" class="icons" />
      Category : {{show.listedIn}}
    </p>
    <p>
      <font-awesome-icon icon="video" class="icons" />
      Type : {{show.type}}
    </p>
    <p>
      <font-awesome-icon icon="comments" class="icons" />
      Description : {{show.description}}
    </p>
  </div>
</div>
```

Figure 13 - Code Snippet For invoking rendering time function

Since this is the vue.js template the list is rendered using the **v-for** directive. When the template is rendering the list and it reaches the last element on the template it fires the **calculateRenderingTime()** method. When this function will be called exactly and what is **6229** in the above figure is explained in the findings section.

```
calculateRenderingTime() {
  var t1 = performance.now();
  console.log("END==>", t1);
  return "";
},
```

Figure 14 - Function for calculating rendering time

In This **calculateRenderingTime()** function, again performance.now() method is called. And if we take the difference between start performance time and end performance time, The time taken to make the AJAX call and render the template will be obtained.

This is how the time is taken to render the real-time data is calculated. This parameter will be understood more clearly in the findings section.

So this is all about the test we are going to perform on the applications built using all the frameworks

Conclusion

This chapter gives an overview of how this research will be carried out, what are the chosen framework & what are the selected metrics. Also, it shows how these metrics will be found out and what are the tools used for finding those metrics. It also talks about The app built for carrying out the performance test in the browser.

CHAPTER 4: IMPLEMENTATION

In the previous chapter, the merits which will be computed to check the performance of the web application are discussed. But to compute those parameters a web application has to be built. So this chapter will discuss how the web application is built using all 4 JavaScript frameworks and it will also discuss the supplementary development needed to support this web application.

This study doesn't focus on the back-end of the web application so not much is discussed about the back-end part of this application, just overview is given.

4.1 Backend Implementation

Backend is comprised of three sections usually a server, application, and a database. There are many languages available for writing code in back-end for example JAVA, PHP, PYTHON, etc. Also for the development of the database, there are various environments available like MySQL, Oracle DB, SQL Server.

4.1.1 Application – Java & REST API

For the back-end application, JAVA is used. Using JAVA multiple REST APIs have been implemented. REST stands for restful. They are also called web services. Webservices are becoming the most integral part of web application development. Not just web development but for android development as well.

4.1.2 Database - MySQL

MySQL is the oldest Database Management System. It s open-source software. It uses SQL as its query language to perform the CRUD operation with the number of the data row & columns. It also has support for stored procedures & functions.

4.1.3 Server

Every Web application needs some kind of web server over which the application is served to the entire world. Tomcat 9 is used for this study. There are other servers available also like Apache server, Node.js server, Nginx server, etc.

4.1.4 Tools & Softwares Used

For the development of JAVA, Eclipse IDE is used. IDE stands for Integrated Development Environment. And for the front-end, Microsofts Visual Studio Code is used. And for database and SQL Navicat premium software is used.

4.1.5 Development & Testing Environment

The system used for Development & testing for this app has the following configurations

OS: Windows 10 Pro.

Processor: intel(r) core(tm) i7-4900mq cpu @ 2.80ghz

RAM: 32 GB

Browser: Google Chrome (84.0.4147.135)

4.2 Front-end Implementation

Four frameworks are chosen for this artifact. And their detailed introduction is given in section 3.2. The UI design of the application built each framework is identical. All the frameworks differ in the code structure, style of rendering the lists, binding of input fields, etc. From all the parameters we have one parameter as **Rendering Time For Real-Time Data**, and a way to calculate this parameter is very different in all the frameworks, so this is explained in depth. Let's see how the 4 applications are built. Throughout the application, we just have one route

as of now and that route is the Home route. Also, the Static Header is implemented using CSS's flex properties.

4.2.1 Angular Implementation.

In angular component is the basic building block of the entire application. Angular 8 is written in TypeScript so components are written in Typescripts as well.

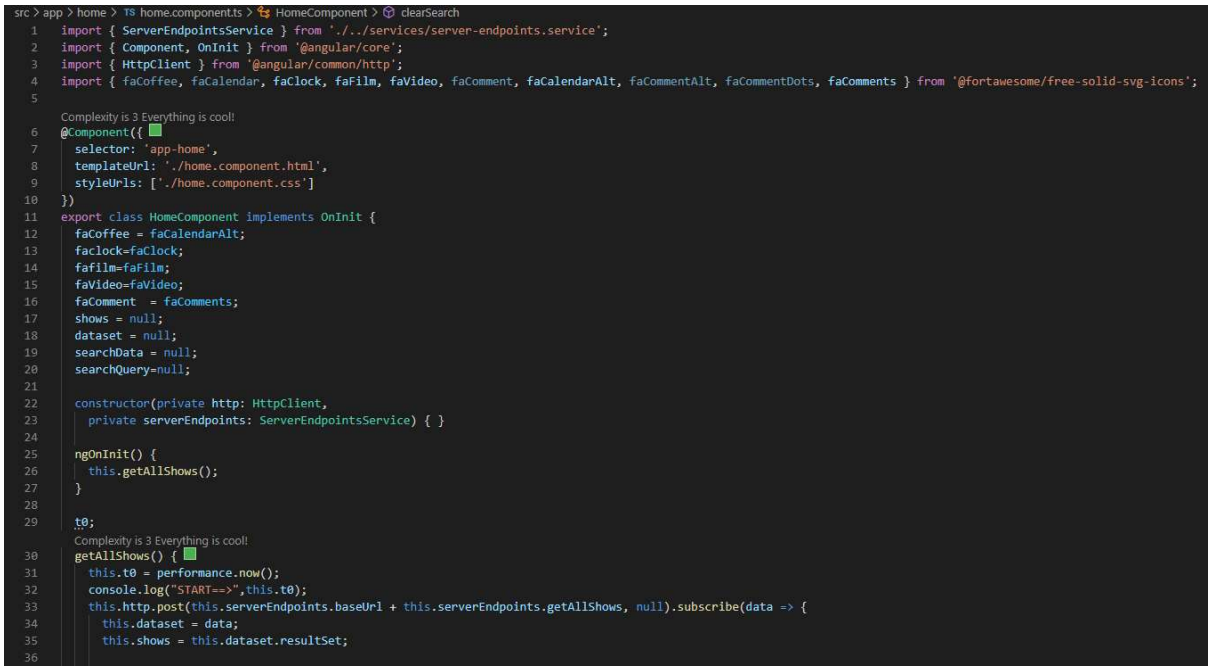


Figure 15 - Angular Component

This is how angular components look like. It has various import statements and then @component annotation and then class.

```
ngOnInit() {
  this.getAllShows();
}

t0;
Complexity is 3 Everything is cool!
getAllShows() {
  this.t0 = performance.now();
  console.log("START==>", this.t0);
  this.http.post(this.serverEndpoints.baseUrl + this.serverEndpoints.getAllShows, null).subscribe(data => {
    this.dataset = data;
    this.shows = this.dataset.resultSet;


  }, error => {

  })
}
```

Figure 16 - Function for getting all the data at load time

When app loads for the first time we have to display some movies and shows retrieved from the database.

So **getAllShows()** function is written in **home.component.ts** file. This function requests to the server for the data using AJAX. And when the data is fetched it is passed to this.shows variable. We want to execute this function as soon as the component is loaded. So as per the Life Cycle hook of the angular framework when the component is loaded whatever is written inside the ngOnInit gets executed therefore this function is called inside the ngOnInit method.

Once the data is retrieved it has to be displayed on the view. Views in angular are called a template.

```
<div class="search-container">
  <input type="text" [(ngModel)]="searchQuery" class="search-input">
  <button (click)="searchSHows()" class="search-button">SEARCH</button>
  <button (click)="clearSearch()" class="search-button">CLEAR</button>
</div>
<div class="flex-container">
  <div *ngFor="let show of shows" class="show-item"  >
    <div *ngIf="show.netflixShowId==100">
      <p *ngIf="show.netflixShowId==100; then calculateRenderingTime()">LAST</p>
    </div>
    <div  class="item-header-container" >
      <h3>{{show.title}}</h3>
    </div>

    <p> <fa-icon [icon]="faCoffee" class="icons"></fa-icon> Year : {{show.yearRelased}}</p>


    <p> <fa-icon [icon]="faclock" class="icons"></fa-icon> Duration : {{show.duration}}</p>
    <p> <fa-icon [icon]="fafilm" class="icons"></fa-icon> Category : {{show.listedIn}}</p>
    <p> <fa-icon [icon]="faVideo" class="icons"></fa-icon> Type : {{show.type}}</p>
    <p> <fa-icon [icon]="faComment" class="icons"></fa-icon> Description : {{show.description}}</p>  </div>

</div>
<!-- [ngStyle]="{'background-color': getRandomBackground()}" -->
```

Figure 17 - Angular template

To display the list on the template the ***ngFor** directive provided by angular is used. These attributes display all the elements inside that list on the webpage.

Also for searching the movies and shows the input filed is given for typing the search query. model is the directive used for two way data binding in angular. Once the user enters the search query and clicks the search button a search shows() functions get executed from the component and the shows and movies based on that search are obtained. And then again that data is passed to the template and the web page is updated.

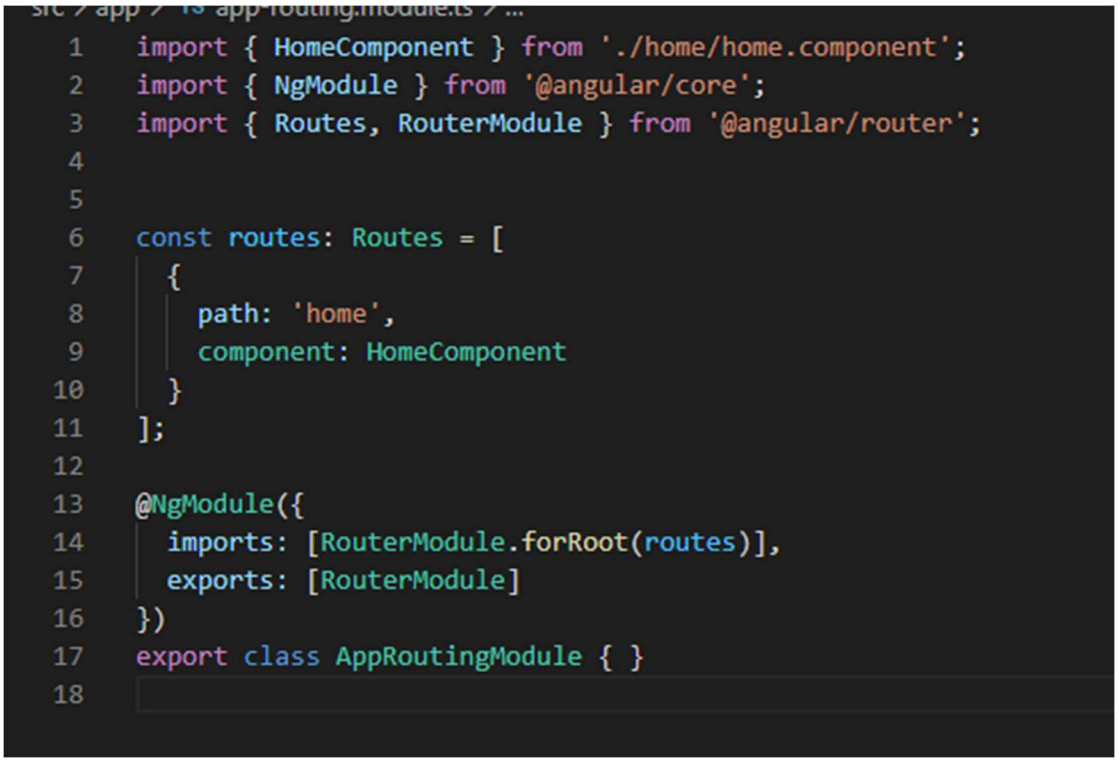
```
calculateRenderingTime() {
  console.log("END==>", performance.now());
}

Complexity is 3 Everything is cool!
searchSHows() {
  this.t0 = performance.now();
  console.log('now');
  console.log(performance.now());

  this.http.post(this.serverEndpoints.baseUrl + this.serverEndpoints.searchSHows, { title: this.searchQuery }).subscribe(data => {
    this.searchData = data;
    this.shows = this.searchData.resultSet;

  }, error => {

  })
}
clearSearch() {
  this.searchQuery = null;
  this.getAllShows();

}
```

Figure 18 - Function for getting results based on a search query

Once the list is updated and the last element is rendered the function **calclulateRenderingTime()** is executed and it just consoles the output. Also for clearing the search input field and getting all the shows which are displayed on the first the **clearSearch()** method is written which will be executed once the clear button is clicked.

Approach to Rendering Time For Real-Time Data:

In angular to achieve this, ***ngIf** directive is used. We can pass the condition to this directive and based on that condition a method can be invoked or CSS could be updated or element is made hidden or visible. So as per shown in image 17 if rendering time has to be calculated on 100^{th} element, and the id of 100^{th} element is known, then the id of the current element will be checked against the known id of 100^{th} element and if condition satisfies then the **calclulateRenderingTime()** function is called and we get the rendering time. This is how the Rendering time of the n^{th} element can be calculated.



```
import { HomeComponent } from './home/home.component';
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';


const routes: Routes = [
  {
    path: 'home',
    component: HomeComponent
  }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Figure 19 - Routing in Angular

This is how routing looks like in angular 8 . Just have to specify the path and component name associated with that path.

4.2.2 ReactJS Implementation:

Like Angular react is also comprises of components.

```
import React, { Component } from "react";
import "./home.css";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import {
  faCalendarAlt,
  faClock,
  faFilm,
  faVideo,
  faComments,
} from "@fortawesome/free-solid-svg-icons";

Complexity is 26 Bloody hell...
class Home extends Component {
  getAllShows = "getAllShows";

  //   state = {
  //     shows: [],
  //   };
  constructor(props) {
    super(props);
    this.state = {
      shows: [],
      searchQuery: "",
    };

    this.searchSHows = this.searchSHows.bind(this);
    this.handleChange = this.handleChange.bind(this);
    this.clearSearch=this.clearSearch.bind(this);
  }

  handleChange(event) {
    console.log(event.target.value);
    this.setState({
      searchQuery: event.target.value,
    });
  }
```

Figure 20 - React Component

In react, the separate template is not maintained only the all the HTML code is written inside the component only. The HTML needed for the page is returned using the **render()** method of react. All this HTML will be sent to the browser and rendered accordingly.

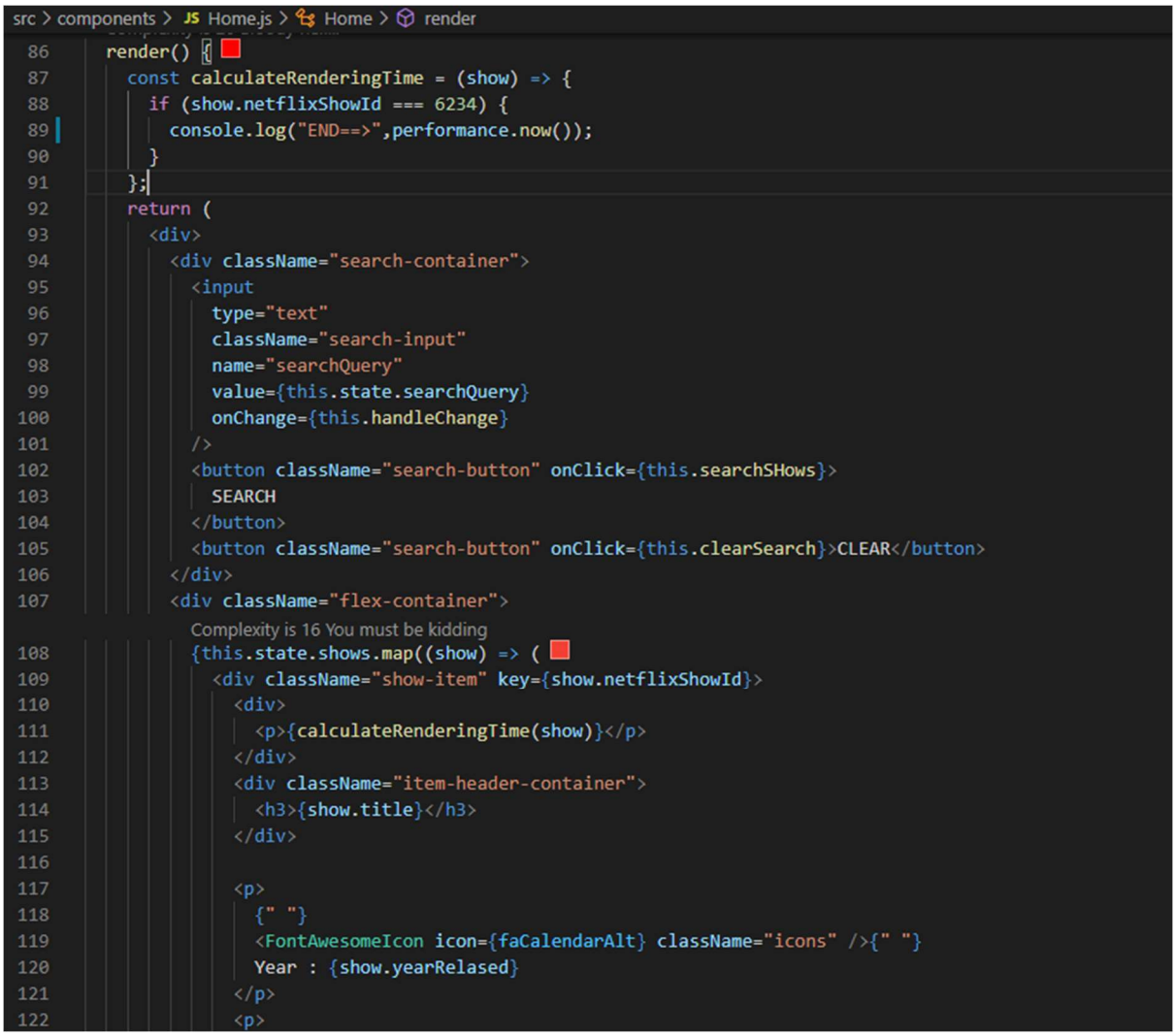


Figure 21 - Template rendering in react

For displaying the list in react the map function is used. The two-way binding used for the input filed is **value={this.state.searchQuery}** also the separate function is written to track the changes in the input field.

Like angular's ngOnInit() react also has the **componentDidMount()** method. Which is evoked when the component is invoked. In that AJAX call is made to fetch the first time data.

```
componentDidMount() {
  console.log(performance.now());
  // POST request using fetch with error handling
  const requestOptions = {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: null,
  };
  fetch("http://localhost:8080/netflix/getAllShows", requestOptions)
    Complexity is 5 Everything is cool!
    .then(async (response) => {
      const data = await response.json();
      this.state.shows = data.resultSet;

      // check for error response
      if (!response.ok) {
        // get error message from body or default to response status
        const error = (data && data.message) || response.status;
        return Promise.reject(error);
      }

      this.setState(this.state.shows);
    })
    .catch((error) => {
      this.setState({ errorMessage: error.toString() });
      console.error("There was an error!", error);
    });
}
```

Figure 22 - React componentDidMount() function

For getting the data based on search query separate AJAX call is written which will be invoked on clicking the search button.

```
searchSHows() {
  const jsonBody = JSON.stringify({ title: this.state.searchQuery });

  console.log(performance.now());
  // POST request using fetch with error handling
  const requestOptions = {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: jsonBody,
  };
  fetch("http://localhost:8080/netflix/searchSHows", requestOptions)
    Complexity is 5 Everything is cool!
    .then(async (response) => {
      const data = await response.json();
      this.state.shows = data.resultSet;
      this.setState({ shows: this.state.shows });
      // check for error response
      if (!response.ok) {
        // get error message from body or default to response status
        const error = (data && data.message) || response.status;
        return Promise.reject(error);
      }

      // this.setState({ postId: data.id });
    })
    .catch((error) => {
      //this.setState({ errorMessage: error.toString() });
      console.error("There was an error!", error);
    });
}

clearSearch() {
  this.setState({
    searchQuery: "",
  });
  this.componentDidMount();
}
```

Figure 23 - Function for getting results based on search query

Approach to Rendering Time For Real-Time Data:

In react when the list is rendering it calls the function to calculate rendering time on every element. It passes the current show or movie to the function and if that movie id is the last then theta rendering function simply prints the log as an End Time.

4.2.3 Vue.js Implementation

In vue.js all the components, template, and styling code .i.e CSS can be written in a single file. This code is written in a file with extension **.vue.** In our case **home.vue.** The code is separated by the tags. All the HTML code is written inside a **<template>..</template>** tag. All Scripting code is written inside **<script>..</script>.** And all the required styling code is written inside a **<style scoped>…</style>.** The scoped directive specifies that the CSS written inside that should be applied to this template only.

```
<template>
  <div>
    <div class="header">
      <div class="logo-container">
        <img src="../assets/logo.jpg" alt class="logo-image" />
      </div>
      <div class="menu-cotianer">
        <span>
          <a href style="color:white">HOME</a>
        </span>
      </div>
    </div>
    <div class="search-container">
      <input type="text" class="search-input" v-model="seacrhQuery" />
      <button class="search-button" @click="searchSHows">SEARCH</button>
      <button class="search-button" @click="clearSearch">CLEAR</button>
    </div>
    <div class="flex-container">
      <div v-for="show in shows" class="show-item" :key="show.netflixShowId">
        <div v-if="show.netflixShowId==6229">
          <p v-bind:style="{ 'background-color': calculateRenderingTime() }">LAST</p>
        </div>
        <div class="item-header-container">
          <h3>{{show.title}}</h3>
        </div>

        <p>
          <font-awesome-icon icon="calendar-alt" class="icons" />
          Year : {{show.yearRelased}}
        </p>
        <p>
          <font-awesome-icon icon="clock" class="icons" />
          Duration : {{show.duration}}
        </p>
        <p>
          <font-awesome-icon icon="film" class="icons" />
          Category : {{show.listedIn}}
        </p>
        <p>
```

Figure 24 - Template in vue.js

The above figure shows how templates are written. It starts with the **<template>** tag. For data binding, the **v-model** directive provided by the venue is used. And to make buttons clickable **@click** is used. For displaying the list **v-for** which is default directive from vue is used.

```
<script>
import axios from "axios";
import serverEndpoints from "./ServerEndpoints.js";
export default {
  //t0:0,
  //seacrhQuery:'',
  name: "Home",
  props: {
    msg: String,
  },
  data() {
    return { shows: [], seacrhQuery: "" };
  },
  mounted() {
    this.getAllShows();
  },
  methods: {
    Complexity is 3 Everything is cool!
    getAllShows() {
      //console.log("START",performance.now());
      axios
        .post("http://localhost:8080/netflix/getAllShows")
        .then((response) => {
          this.shows = response.data.resultSet;
        })
        .catch((err) => {
          // Manage the state of the application if the request
          // has failed
          console.log(err);
        });
    },
    getRandomBackground() {
      return serverEndpoints.cssGradients[Math.floor(Math.random() * 20)];
    },
    calculateRenderingTime() {
      var t1 = performance.now();
      console.log("END==>", t1);
      return "";
    },
```

Figure 25 - Component in vue.js

This is how components look likes in the vue. Like **ngOnIniti()** and **componentDidMount()** methods from angular and react respectively,vue.js have the **mounted()** method which gets invoked when the component is instantiated and when it is instantiated the **getAllMethod()** is

called and all the data is fetched from the server. Similarly, **getSearchShows()** is written to get data based on the search query. All the required methods are written inside the method object.

Approach to Rendering Time For Real-Time Data:

The approach used in Vue.js & Angular 8 is very similar. Here **v-if** is used to check the current elements show id with the last elements show id. If this condition satisfies the element under that condition will be rendered and if that element is rendered it will call a **caluclateRenderingTime()** method for setting a CSS style using the **v-bind** property as shown in figure 24.

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Home from '../components/Home.vue'
Vue.use(VueRouter)

  const routes = [
  {
    path: '/home',
    name: 'Home',
    component: Home
  }
]

const router = new VueRouter({
  mode: 'history',
  routes
})

export default router
```

Figure 26 - Routing in Vue.js

Routing in vue is provided as shown in the above image. It takes the const routes as the array of an object where each object represents the separate route. This object has a path, name & component fields.

4.2.4 Ember.js Implementation

In ember.js component and template is written in the separate file. The component code is written in the home.js file. The **getAllShows()** method is called in the constructor of a class. The basic functionality of the constructor is to initialize the properties of its class. In the same way, get **getSearchShows()** is written which will be called once the user clicks on the search button, and template will be updated.

```
/* eslint-disable ember/no-jquery */
import Controller from "@ember/controller";
import { action } from "@ember/object";
//import jQuery from 'jquery';
import Ember from "ember";
import jQuery from "jquery";
import { tracked } from "@glimmer/tracking";
import { all } from "@fortawesome/free-solid-svg-icons";
export default class HomeController extends Controller {
  // const element = jQuery('#special');
  constructor(...args) {
    super(...args);
    this.isLarge1 = true;
    //this.getAllShows();
    this.getAllShows();
  }

  mystyle = "background-color:yellow";
  @tracked searchQuery="";
  @tracked shows = [{ name: "djbnjdfbdfb" }];

  @action getAllShows() {
    jQuery.ajax({
      url : 'http://localhost:8080/netflix/getAllShows',
      type: 'POST',
      data: null,
      headers: {
            "Content-Type": "application/json"
        },
      success : this.setSHows
  })
  }

  @action handleData(data /* , textStatus, jqXHR */) {
    //this.shows=data.resultSet.splice(2, 6000);
    this.shows = data.resultSet;
    //do some stuff
  }
```

Figure 27 - Ember.js Controller

The extension of the template is **.hbs,** in our case **home.hbs.**

.

```
<div class="search-container">
  {{!-- <input type="text"  class="search-input"  value="searchQuery"> --}}
  {{input type="text" class="search-input" value=searchQuery }}

  <button class="search-button" {{action 'searchShow'}}>SEARCH</button>

  <button class="search-button" {{action clearSearch}}>CLEAR</button>
</div>
<div class="flex-container">
  {{#each shows as |show|}}
  <div class="show-item">
    <div class="item-header-container">
      <h3>{{show.title}}</h3>


    </div>

    <p>
      <FaIcon @icon="calendar-alt" class="icons" /> Year : {{show.yearRelased}}</p>
    <p>
      <FaIcon @icon="clock" class="icons" /> Duration : {{show.duration}}</p>
    <p>
      <FaIcon @icon="film" class="icons" /> Category : {{show.listedIn}}</p>
    <p>
      <FaIcon @icon="video" class="icons" /> Type : {{show.type}}</p>
    <p>
      <FaIcon @icon="comments" class="icons" /> Description : {{show.description}}</p>
    <p> {{homehelper show.netflixShowId}}</p>
    {{!--
<p> {{show.yearRelased}}</p>
    <p>{{show.duration}}</p>
    <p>{{show.listedIn}}</p>
    <p>{{show.type}}</p>
    <p>{{show.description}}</p>
    <p> {{homehelper show.netflixShowId}}</p> --}}
  </div>
  {{/each}}

</div>
```

Figure 28 - Ember.js Template

For looping through the list ember's **each** helper is used. **{{#each shows as |show|}} ... {{/each}}.** Whatever HTML is written inside this each tag will be repeated the number of time shows is there. For the input binding, the custom **{{input }}** element is used. And **value** is used for binding the input value. For an action to execute on button click **{{action 'searchShow'}}**

is used. On clicking, this search show function will be called and selected data will be retrieved from the Database.

Approach to Rendering Time For Real-Time Data:

In ember.js helper is implemented to achieve this functionality. Helper is nothing but JavaScript function which can be invoked from the template as shown in image 28. A current show is passed as an argument and it is compared in the helper with the last element's id.

```
import { helper } from '@ember/component/helper';

Complexity is 3 Everything is cool!
export default helper(function homehelper(show) {
  //console.log(show);
  if(show[0]==6229){
    console.log("END==>",performance.now());
  }
  return "";
});
```

Figure 29 - Ember.js helper function for calculating end time

Conclusion

This chapter has provided an overview of how the Netflix movie app has been implemented using all the four selected framework. It mentions how these applications have different components, how their data-binding is implemented, and how the primary data is called as soon as the application is instantiated and how routing is implemented. Also the development environment, technologies used in the back-end, and tools & software used.

CHAPTER 5: FINDINGS & DISCUSSIONS

This chapter will discuss the result of all the metrics using all for frameworks.

5.1 LOC:

To measure the lines of code we have to run two commands with the directory as its parameter. for example, if all the code is in the src folder, then the command will be executed in the following way. We will take readings by using two tools for greater accuracy and their average will be taken out

```
C:\Local Disk D\STUDY\Thesis\CODE\FRONT-END\thesis-netflix-angular8\angular>sloc src
```

Figure 30 - command for counting LOC

The below image shows all the readings taken by these two tools for all the 4 frameworks.

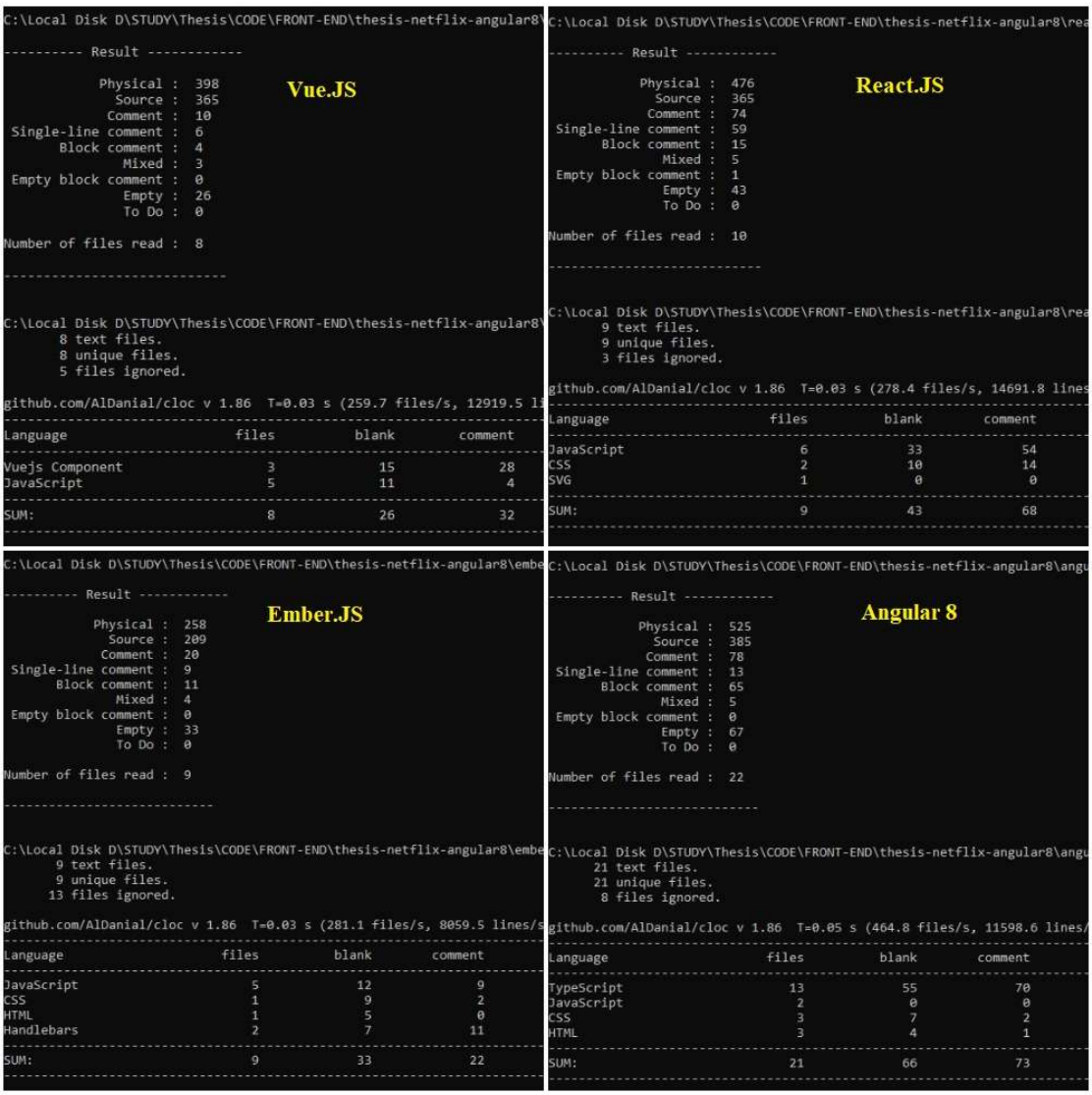


Figure 31 - LOC readings

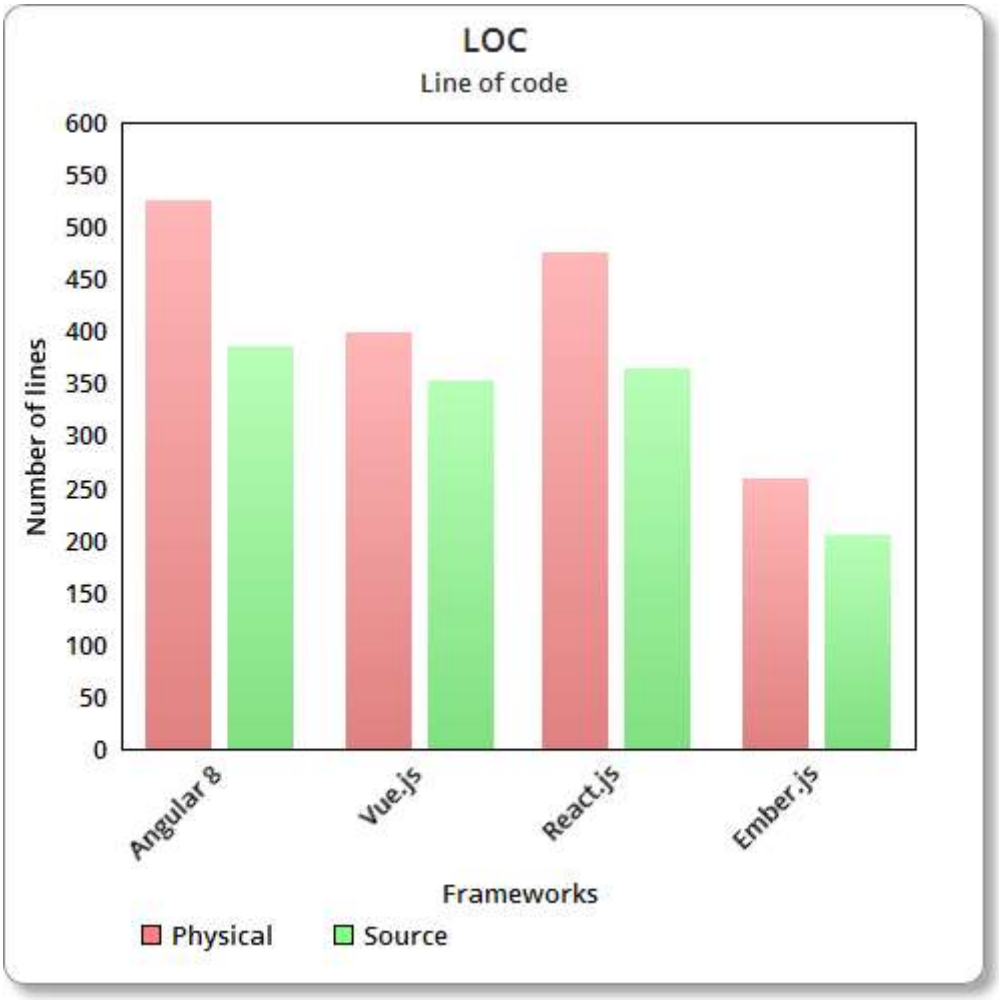


Figure 32 - Graph For LOC

From the image 31 and graph 32, a visual comparison of lines of codes can be seen. Angular 8 takes the highest number of lines to create the web application. And on the other side ember.js has the least amount code in both its physical code & logical code. After angular react takes the 2nd place for highest LOC's in both SLOC & LLOC.

5.2 First Contentful Paint

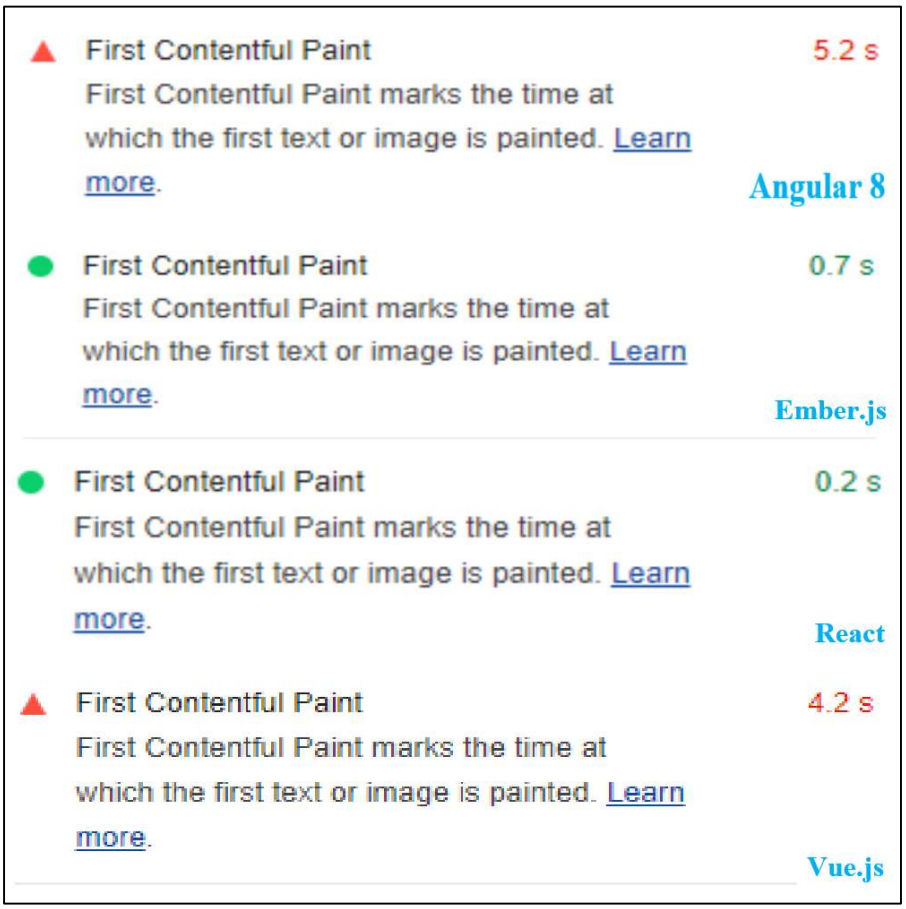


Figure 33 - Readings for FCP

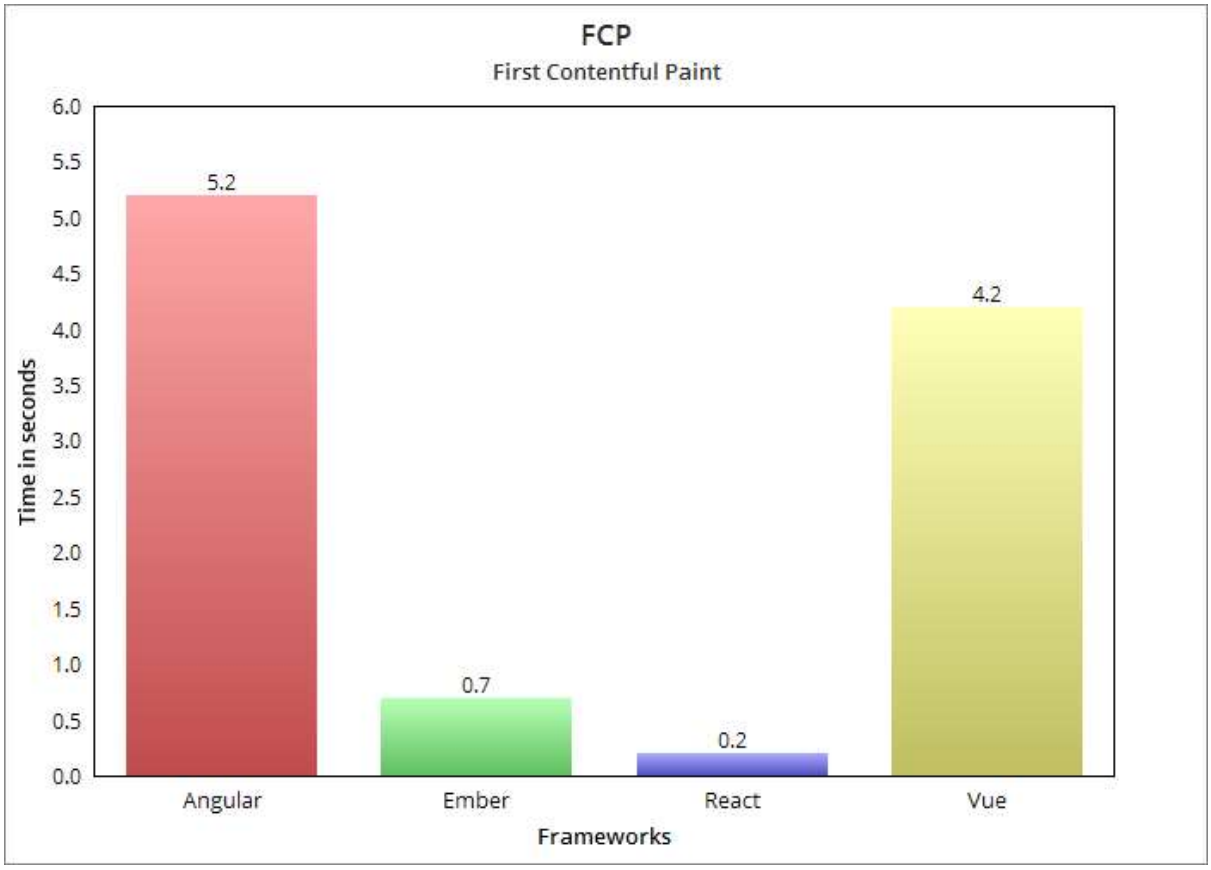


Figure 34 - Graph for FCP

The chart shown above represents the FCP.

For showing the first content on the webpage react takes the least amount of time which 0.2s. Angular 5.2s being the highest takes significantly much more time than the react. Vue js also take 4.2s which is the second-highest and ember js shows the efficient timing of 0.7s.

5.3 Speed Index:

SI shown in image 35 is in the following order:

1. Angular
2. Ember
3. React
4. Vue

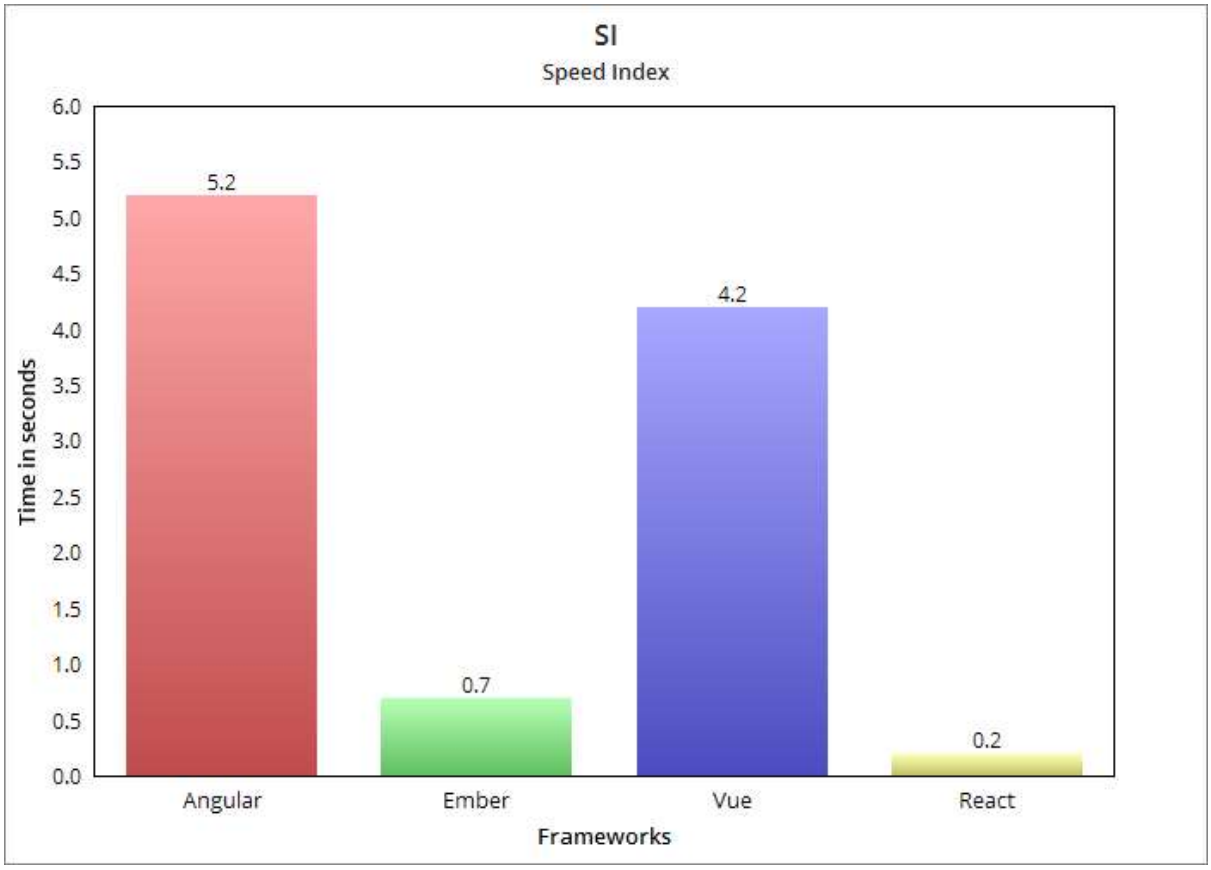


Figure 35 - Graph for SI

Lower the speed index is better is the performance of the application. The application built using the react js shows the best speed index of 0.2s where applications built using angular has the worst speed index of 5.2s. Ember.js and react.js show the speed index of 0.7 and 0.2 respectively.

5.4 Time To Interactive

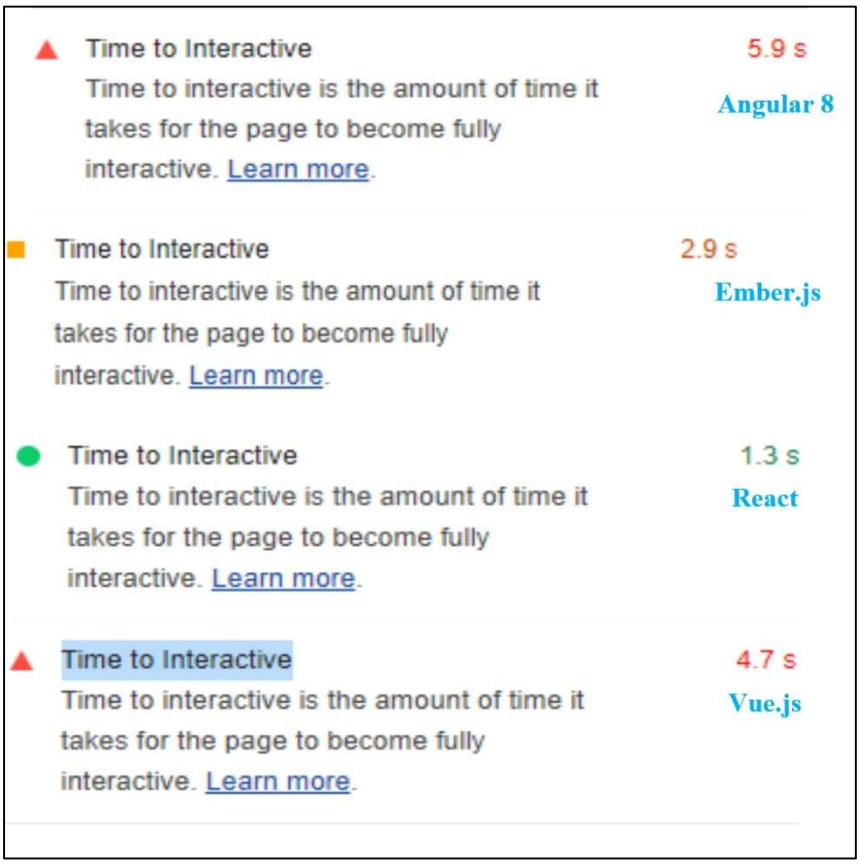


Figure 36 - Reading for TTI

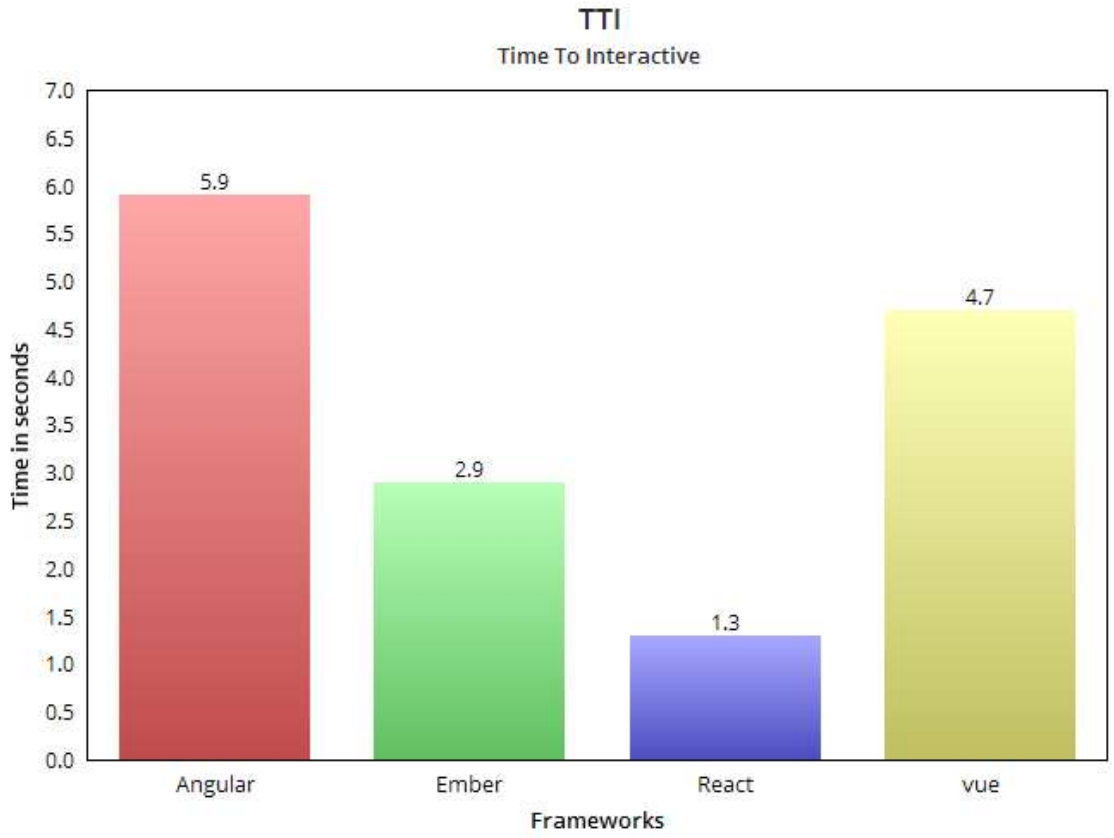


Figure 37 - Readings for TTI

It is very obvious that the page whose speed index is high, their TTI will also be high. SI of angular and vue.js is very poor therefore their TTI is also poor compared to the Ember and react.js. TTI of Angular is almost 6 seconds while TTI of React is 1.3s.

5.5 Total Blocking Time

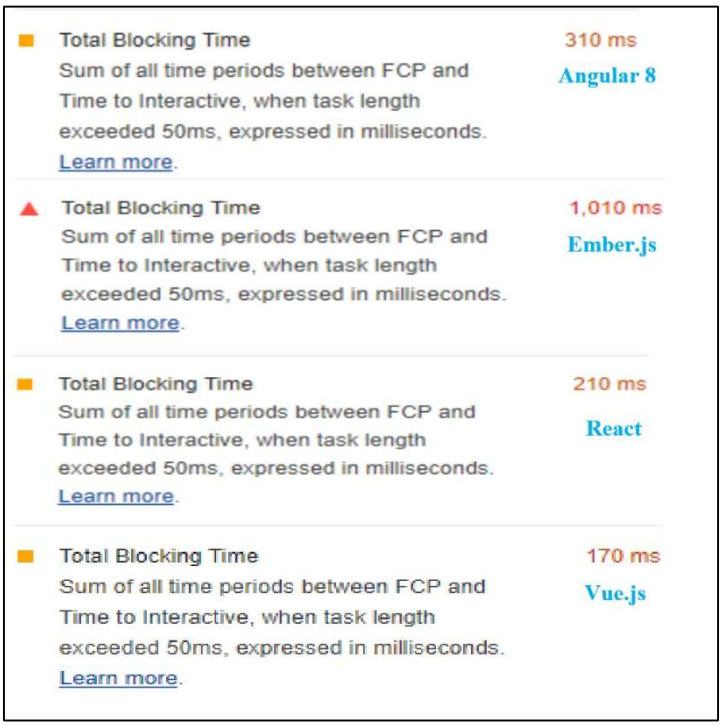


Figure 38 - reading for TBT

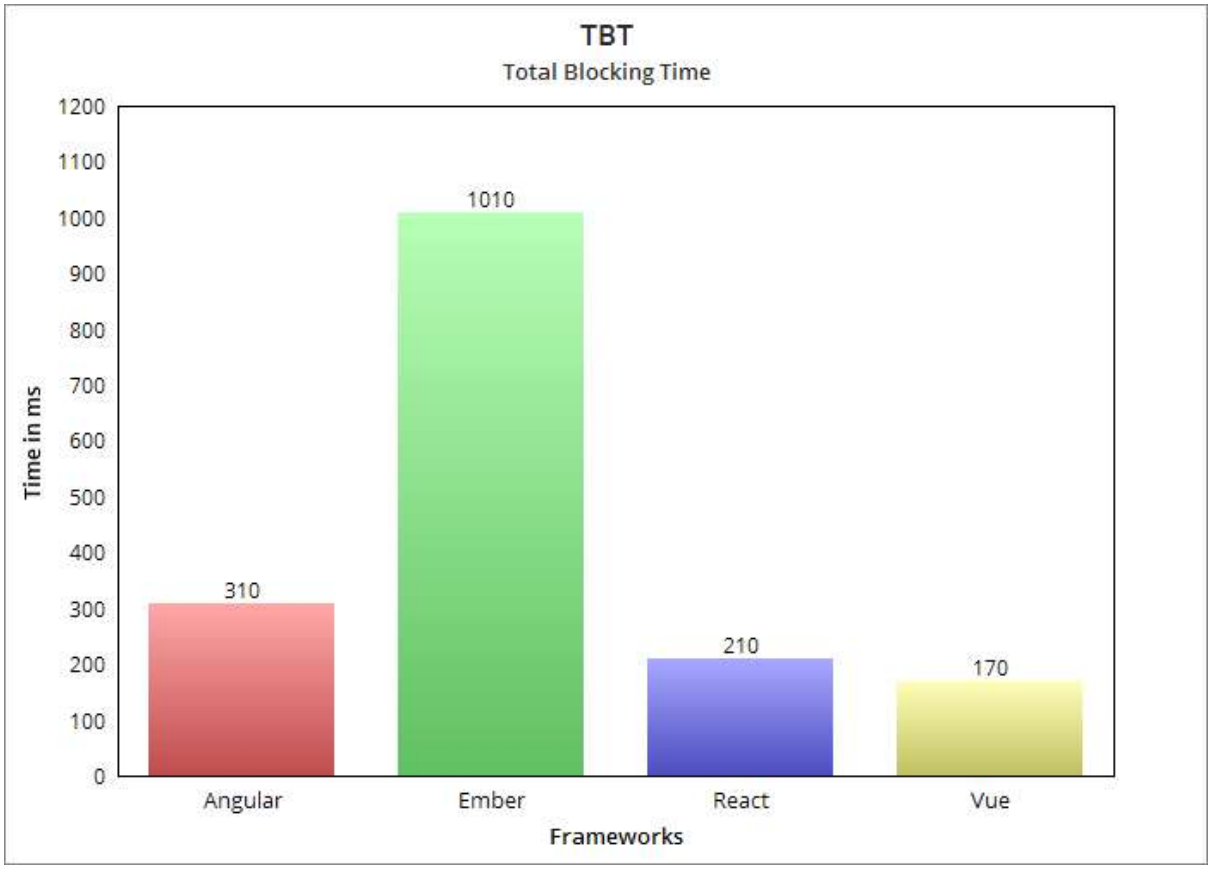


Figure 39 - Graph for TBT

TBT for Ember 1010 ms application way higher than the rest of the applications. Vue.js have the lowest TBT of 170 ms followed by reacting and Angular 210 ms and 310 ms respectively.

5.6 Time Taken To Render The Real-Time Data:

For more accuracy, this test will be calculated thrice in every framework. Once the test is run it will print out the output time in the console as start and end time, then have to take the difference between the start and end time, and after that whatever the value we get will be the Time taken to render the real-time data.

When the user clicks on the search button a timer will be initialized and once the last elements are displayed the end time function will be executed and their difference is the time taken. The technique for calling the end time function is different in each framework.

Figure 40 - UI for the search functionality

On typing kids in the search section and clicking the search button, 339 records will be returned from the server. When the 339th record is displayed on the page the **calculateRenderingTime()** method will be called. How this method is called is different for every framework and the same is explained in the implementation section of each framework.

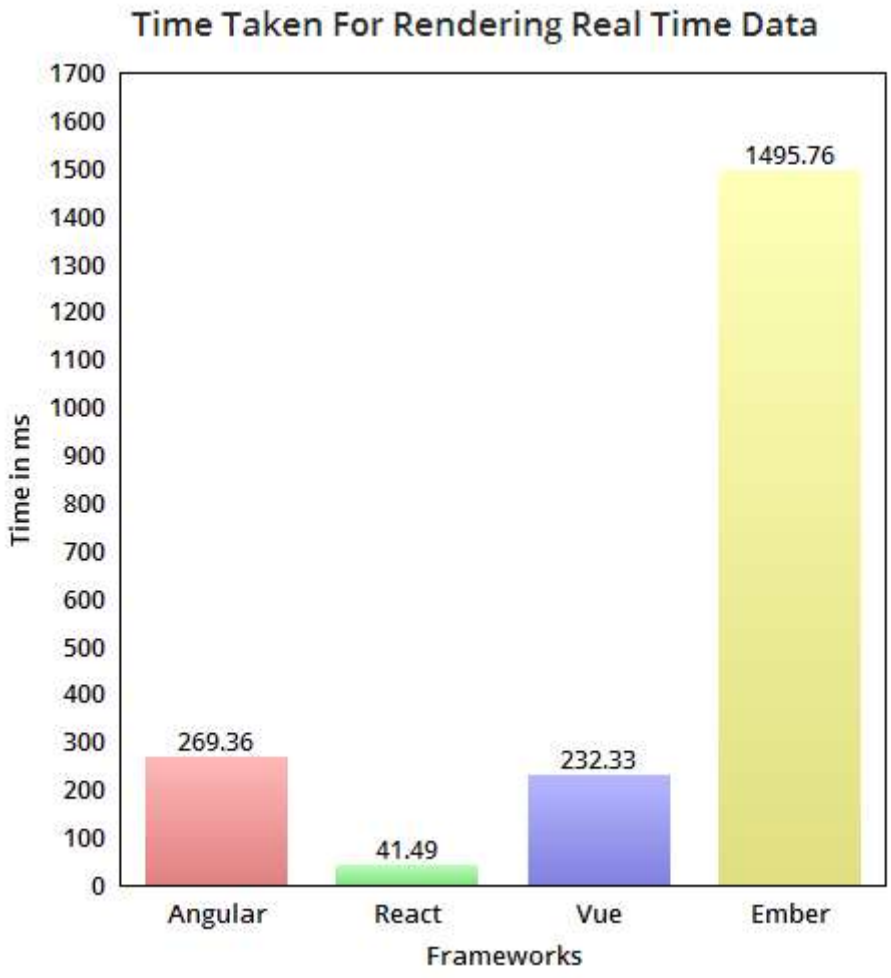


Figure 41 - Graph for list Rendering time

From the figure, one can depict that Ember js takes relatively much higher time (1495.76 ms) to render the list of 339 items. It takes almost 36 times higher time than that of react.js which is the least time i.e 41.49 ms. Angular & Vue.js takes 269.36 ms and 232.33 ms respectively which is much lesser than Emberjs.

Conclusion

This section gives an overview of the findings found after running all the performance tests. Overall the speed & interactivity of apps built using the Angular framework is not that good compared to the other frameworks. On the other hand, apps built using the react frameworks perform well in terms of speed and interactivity. EmberJs also performs well when it comes to speed. If we talk about the LOC aspect of the framework. Angular & React needs a large amount of code compared to vue and emberjs. EmberJs needs the least amount of code. So all in all emberjs performs well in the browser with less amount of code.

CHAPTER 6: CONCLUSION & FUTURE SCOPE

Every user likes the application which gives them a better user experience. Better in terms of the loading speed of the page how smooth it behaves etc. to achieve this better performance there are several frameworks available to use. Therefore This study has presented an effective approach to compare & evaluate the performances of the JavaScript frameworks for single-page applications on the browser. This study also has pointed out the Importance & need of the single page application to cope up with the fast-growing internet and related technologies. This study has also described the past work who has tried to evaluate the different frameworks. But the evaluation metrics and frameworks are chosen are different. In previous studies, the major focus was on the framework's code & complexity and not how this framework performs. Those researchers tried to evaluate the Code complexity, Lines of Code, Maintainability index, Code size, etc. But this study has talked about how to compare the performance of frameworks in the browser. This study evaluates different parameters related to speed like Speed Index, FCP, TBT, TTI, Time taken to render the list, etc. This study revealed that even if writing the code is much easier using an angular framework but their performance on the browser is poor compared to the other frameworks also it takes more LOC to build the app. Also, apps built using react are the fastest to display on the browser and eventually they become interactive earlier than other frameworks. The efficiency of the app built using the ember is also good in terms of speed and loading time.

Future Scope

The application developed in this study looks to be good and reliable for browser performance. In the future, it can be made to work well on other platforms like mobile & tablets, etc. and this framework’s performances can be compared in those platforms as well.

Additionally, more features can be added to the application like filtering, sorting, pipes, etc. and their performances can be evaluated.

This study expects that future developer will extend this application further and more parameters will be evaluated.

CHAPTER 7: REFERENCES

Bainczyk, A. *et al.* (2017) 'Model-Based Testing Without Models: The TodoMVC Case Study', in Katoen, J.-P., Langerak, R., and Rensink, A. (eds) *ModelEd, TestEd, TrustEd: Essays Dedicated to Ed Brinksma on the Occasion of His 60th Birthday*. Cham: Springer International Publishing, pp. 125–144. doi: 10.1007/978-3-319-68270-9_7.

Bouckaert, S. *et al.* (2011) 'Benchmarking computers and computer networks', *EU FIRE White Paper*, (i), pp. 1–14. Available at: http://www.ict-fire.eu/news/view/article/benchmarking-computers-and-computer-networks-white-paper.html.

Budiman, E. *et al.* (2018) 'Performance analysis of the resource loading time for borneo biodiversity information system', *Proceedings of the 3rd International Conference on Informatics and Computing, ICIC 2018*. IEEE, pp. 5–9. doi: 10.1109/IAC.2018.8780515.

Dale, T. (2012) *Our {Approach} to {Routing} in {Ember}.js - tomdale.net*. Available at: https://tomdale.net/2012/05/ember-routing/ (Accessed: 20 August 2020).

Davila, H. F. and Navon, J. (2015) 'Performance of JavaScript Frameworks on Web Single Page Applications (Spa)'.

Dodds, K. C. (2020) 'kentcdodds/cloc'. Available at: https://github.com/kentcdodds/cloc (Accessed: 21 August 2020).

Farwell, M. (2017) 'Yakov Fain on Angular', *IEEE Software*, 34(6), pp. 109–112. doi: 10.1109/MS.2017.4121218.

Fowler, S., Denuzière, L. and Granicz, A. (2015) 'Reactive Single-Page Applications with Dynamic Dataflow', in Pontelli, E. and Son, T. C. (eds) *Practical Aspects of Declarative Languages*. Cham: Springer International Publishing, pp. 58–73.

Furtak, M. and Wittie, M. P. (2019) 'The ODDness of webpages', *TMA 2019 - Proceedings of the 3rd Network Traffic Measurement and Analysis Conference*, pp. 33–40. doi: 10.23919/TMA.2019.8784552.

Gao, Q., Dey, P. and Ahammad, P. (2017) 'Perceived performance of Top Retail webpages in the wild: Insights from large-scale crowdsourcing of above-the-fold QoE', *Internet QoE 2017 - Proceedings of the 2017 Workshop on QoE-Based Analysis and Management of Data Communication Networks, Part of SIGCOMM 2017*, pp. 13–18. doi: 10.1145/3098603.3098606.

Gizas, A. B., Christodoulou, S. P. and Papatheodorou, T. S. (2012) 'Comparative evaluation of JavaScript frameworks', *WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web Companion*, (Cc), pp. 513–514. doi: 10.1145/2187980.2188103.

Google (2020) '{GoogleChrome}/lighthouse'. GoogleChrome. Available at: https://github.com/GoogleChrome/lighthouse (Accessed: 21 August 2020).

Graziotin, D. and Abrahamsson, P. (2013) 'Making Sense Out of a Jungle of JavaScript Frameworks', pp. 334–337. doi: 10.1007/978-3-642-39259-7_28.

Guard, K. (2020) 'Why {Page} {Speed} {Matters} and {How} to {Improve} {It}', *seoClarity*. Available at: https://www.seoclarity.net/resources/knowledgebase/why-page-speed-matters-16167/ (Accessed: 21 August 2020).

da Hora, D. N. *et al.* (2018) 'Narrowing the Gap Between QoS Metrics and Web QoE Using Above-the-fold Metrics', *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10771 LNCS, pp. 31–43. doi: 10.1007/978-3-319-76481-8_3.

Irish, P. (2020) 'paulirish/speedline'. Available at: https://github.com/paulirish/speedline

(Accessed: 21 August 2020).

Jadhav, M. A. *et al.* (2015) 'Single Page Application using AngularJS', *International Journal of Computer Science and Information Technologies*, 6(3), pp. 2876–2879. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.736.4771&rep=rep1&type=pdf.

Javeed, A. (2019) 'Performance Optimization Techniques for ReactJS', *Proceedings of 2019 3rd IEEE International Conference on Electrical, Computer and Communication Technologies, ICECCT 2019*. IEEE, pp. 0–4. doi: 10.1109/ICECCT.2019.8869134.

Jayakody, A. *et al.* (2017) 'Efficient and Platform Independent CLI Tool for API Migration'.

John, S. (2010) 'CHAT APP WITH REACT JS AND FIREBASE Department of Information Technology'.

Johnson, R. E. (1997) 'Frameworks = (Components + Patterns)', *Communications of the ACM*, 40(10), pp. 39–42. doi: 10.1145/262793.262799.

Khurana, P. and Kumar, D. (2019) 'Growth Analysis of Social, Mobile and Mobile Social Users through Internet', *Proceedings of the International Conference on Machine Learning, Big Data, Cloud and Parallel Computing: Trends, Prespectives and Prospects, COMITCon 2019*. IEEE, pp. 564–567. doi: 10.1109/COMITCon.2019.8862226.

Kohlhase, M. (2020) 'flosse/sloc'. Available at: https://github.com/flosse/sloc (Accessed: 21 August 2020).

Lawrence, C. (2017) 'Benchmarking JavaScript Frameworks', pp. 31–33. doi: 10.21427/D72890.

Le, A. T. (2020) 'Developing a web application for task management with ReactJS'.

Lin, B. *et al.* (2012) 'Comparison between JSON and XML in Applications Based on AJAX',

Proceedings - 2012 International Conference on Computer Science and Service System, CSSS 2012. IEEE, (February 1998), pp. 1174–1177. doi: 10.1109/CSSS.2012.297.

Mardan, A. (2014) 'Todo App', in *Pro Express.js*. Berkeley, CA: Apress, pp. 223–248. doi: 10.1007/978-1-4842-0037-7_20.

Mesbah, A. and Deursen, A. Van (2007) 'Migrating Multi-page Web Applications to Single-page A JAX Interfaces'.

Molin, E. (2016) 'Comparison of Single-Page Application Frameworks: A method of how to compare Single-Page Application frameworks written in JavaScript.', *Degree Project Computer Science and Engineering*.

Moreno, J. and Robles, G. (2015) 'Automatic detection of bad programming habits in scratch: A preliminary study', *Proceedings - Frontiers in Education Conference, FIE*. IEEE, 2015-Febru(February), pp. 23–26. doi: 10.1109/FIE.2014.7044055.

Osmani, A. (2019) 'Are long {JavaScript} tasks delaying your {Time} to {Interactive}?' Available at: https://web.dev/long-tasks-devtools/ (Accessed: 21 August 2020).

Park, R. E. (1992) 'Software Size Measurement : A Framework for Counting Source Statements. In: Technical Report CMU/SEI-92-TR-20.', (September), p. 242. Available at: http://www.sei.cmu.edu/reports/92tr020.pdf.

Rahman, I. A. and Ikbal, I. (no date) 'OPTIMIZATION OF BLAST COMPUTE SITE THROUGH DESIGNING LITESPEED CACHE USING PPDIOO METHOD'.

Ratanaworabhan, P., Livshits, B. and Zorn, B. (2010) 'JSMeter: Comparing the behavior of JavaScript benchmarks with real web applications', *Conference on Web applications*. Available at: http://www.usenix.org/event/webapps10/tech/full_papers/Ratanaworabhan.pdf.

Rentrop, J. *et al.* (2006) 'Software Metrics as Benchmarks for Source Code Quality of Software

Systems', *Measurement*.

Richards, G. *et al.* (2010) 'An analysis of the dynamic behavior of JavaScript programs', *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, (June), pp. 1–12. doi: 10.1145/1806596.1806598.

Romle, A. N. *et al.* (2019) 'Guidelines for Mobile Web Application', *2019 6th International Conference on Research and Innovation in Information Systems (ICRIIS)*, pp. 0–5.

Saif, D., Lung, C.-H. and Matrawy, A. (2020) 'An Early Benchmark of Quality of Experience Between HTTP/2 and HTTP/3 using Lighthouse'. Available at: http://arxiv.org/abs/2004.01978.

Schmidt, D. C. and Buschmann, F. (2003) 'Patterns, frameworks, and middleware: their synergistic relationships', in *25th International Conference on Software Engineering, 2003. Proceedings.*, pp. 694–704. doi: 10.1109/ICSE.2003.1201256.

sean, work (2011) 'How {Loading} {Time} {Affects} {Your} {Bottom} {Line}', *Neil Patel*. Available at: https://neilpatel.com/blog/loading-time/ (Accessed: 21 August 2020).

Seltzer, M. *et al.* (no date) 'The Case for Application-Specific Benchmarking'. IEEE.

Shen, J., Sun, G. and Li, Y. (2018) 'Design and Implementation of E-Commerce Platform Based on Android', *Journal of Computer and Communications*, 06(08), pp. 92–100. doi: 10.4236/jcc.2018.68007.

Shrestha, S. (2015) 'Ember . js front-end framework – SEO challenges and frameworks comparison', (October), pp. 1–47.

Smith, K. (no date) 'Simplifying Ajax-style Web development'. IEEE, pp. 98–101.

Song, J., Zhang, M. and Xie, H. (2019) 'Design and implementation of a Vue.js-based college

teaching system', *International Journal of Emerging Technologies in Learning*, 14(13), pp. 59–69. doi: 10.3991/ijet.v14i13.10709.

Studiengang Bachelor, im *et al.* (no date) 'Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung'.

TodoMVC (no date) 'TodoMVC'. Available at: http://todomvc.com (Accessed: 23 August 2020).

Voutilainen, J. (2017) 'Evaluation of Front-end JavaScript Frameworks for Master Data Management Application Development', (December), pp. 4–6. Available at: https://www.theseus.fi/bitstream/handle/10024/138668/Voutilainen_Jaakko.pdf?sequence=1.

Walton, P. (2019) 'Largest Contentful Paint (LCP)'. Available at: https://web.dev/lcp/#largest-contentful-paint-defined (Accessed: 21 August 2020).

Wang, Q. *et al.* (2008) 'An automatic approach to reengineering common website with AJAX', *Proceedings - International Conference on Next Generation Web Services Practices, NWeSP 2008*, pp. 185–190. doi: 10.1109/NWeSP.2008.34.

CHAPTER 8: APPENDICES

All the code & results are available on the GitHub as a public repository.

https://github.com/saurabh-github-1995/Thesis