A Front-End User Interface Layer Framework for Reactive Web Applications

Zhixong Xiao, Chandana Prasad Withana, Abeer Alsaadoon, Amr Elchouemi

School of Computing and Mathematics, Charles Sturt University, Sydney, Australia
Walden University, USA

Abstract: Nowadays, people are relying more and more on web applications, such as Gmairs, Google Map and Google Docs to complete their daily tasks. However, web applications often fail to provide reactive interactions with users. This paper explores the issues and problems of current web application frameworks and narrowed the research to the User Interface (UI) layer as it is the most important component to focus on in terms of increasing web application reactivity. By integrating two Javascript libraries, namely, Preact and Preact-router into the UI layer, the proposed approach optimizes the way how the web server and web client communicates, which leads to a more reactive web application. The proposed UI layer framework was tested against a current framework and found that the proposed framework reduced a significant amount of page load time. In addition, number of requests sent to a web server was also reduced compared to the current framework. The proposed UI layer framework can be applied to business web applications to increase their applications load time and reactivity. By making their web applications more reactive, it would potentially have a positive impact on the conversion rates of their businesses.

Keywords: Web Application, Front-End User Interface Layer Framework, Web Application Frameworks

Introduction

The fast paced development in network bandwidth and internet technology has pushed web applications to a dominant position. Nowadays, it is possible to use an application anywhere and anytime as long as you are connected to Internet. Due to the convenience provided by web applications, online users start to embrace them in their daily tasks (Nations, 2016). For example, Google Maps services are used worldwide by around 41% of Internet users via their browsers (Privat, 2014). There are about one billion users are using Google Maps each month. Furthermore, Gmail also has more than 1 billion monthly active users according to Techcrunch (Lardinois, 2016).

However, page reactivity has been a major issue in web applications. It is reported that a delay of 100 milliseconds in website load time can reduce 7 percent in conversion rate (Formack, 2017). Another report from BBC highlighted that a half second difference in page load times can lead to a 10% difference in online sales.

The persistent reactivity issue is due to the fact that web application is traditionally hosted on a web server in one place and rendered by a web client in another place (MF, 2017). Therefore, the frequent communications between the two ends lead to a poorer user experience compared to client applications. Some of the background information regarding how web applications works is presented in Fig. 1.

Though security, scalability, maintainability and development speed are important factors that should be considered when building web applications, this paper will only focus on analyzing the current web development framework solutions for addressing the reactivity issue and identifying the best one. The ultimate goal of this paper is to propose a new UI layer framework which can improve the reactivity of web applications.
Despite many researches on improving the reactiveness of web applications, there are still rooms to optimize. Some researchers used AngularJS framework to increase the reactiveness by introducing front-end routing (Chansuwath and Senivongse, 2016; Nikolić et al., 2016). This approach did increase the overall reactiveness of web applications. However, AngularJS framework itself is notoriously heavy and takes a long time to load up. Balasubramanee et al. (2013) proposed to use Bootstrap framework in the front end, however, this approach still relies on the traditional client-to-server paradigm. Therefore, it requires frequent communications between client and server sides in order to request and receive documents (Anderson, 2017). Ahlawat (2016) and Priefer (2014) integrated CMS frameworks into the front-end side in an effort to streamline web application development process, however, the CMS-based solutions are highly inflexible because it depends excessively on third party plugins and add-ons.

The current solutions did not address the web application reactiveness from the UI layer. Therefore, a new solution that focuses on the UI layer is urgently needed.

Load time has been proved to have a significant impact on business. According to Dooley (2012), a mere one-second delay in page load time was accompanied by a 7% decline in sales. In another study, Ancestry claimed a 7% positive rise in conversions after improving the render time of web pages by 68%, whereas AliExpress reduced load time for their pages by 36% and recorded a 10.5% increase in orders and a 27% increase in conversion rates for new customers (Anderson, 2017). Web application users are more satisfied if web applications can provide them with a smooth experience. Therefore, a more reactive solution to the web application can significantly increase conversion rate for online business.

Literature Review

The existing web application frameworks will be divided into three categories based on their technical components. The first section will talk about MVC frameworks, which follows by CMS frameworks and lastly plug-in frameworks will be examined.

MVC Frameworks

For the purpose of building web applications, several web application frameworks have been introduced by various researchers. In efforts to build a secure and scalable web application, Panchal (2016) proposed a framework using JavaServer Pages and Spring as back-end technologies. Though this approach provides a secure back-end for web applications, it focuses mainly on the back-end and did not provide a reactive solution on the front-end. Other researchers presented a similar idea but used different programming languages and frameworks, such as Ruby on Rails or PHP (Meenakshi, 2015; Vohra, 2014; Safronov and Winesett, 2014). Therefore, despite its outstanding
scalability and security, this approach fails to provide a reactive front-end solution.

Some researchers introduced AngularJS framework to address the issue of web application reactivity (Chansuwath and Senivongse, 2016; Nikolić et al., 2016). Similarly, Rahman and Chitra (2015) presented a solution fusing AngularJS with Joomla for building reactive web applications. Furthermore, Balsubramanee et al. (2013) proposed to combine Bootstrap and AngularJS in an effort to expedite the web application development process. However, AngularJS is itself a complex and resource-consuming framework and is not suitable for light-weight web applications. Though the AngularJS framework is not an optimal choice, it seems that the front-end routing technologies used in the framework could be used in combination of other technologies to improve the reactivity of web applications.

Song (2014) proposed to use Ajax in MVC for web application development. Whist this improves the user experience as data exchange between web client and web server is done silently, it still relies on network condition. Pop and Altar (2014) utilized a MVC model for rapid prototyping when building a web application. This approach can boost up the speed of the web application development, however, it fails on the front end to make the web application reactive.

**CMS Frameworks**

Priefer (2014) integrated a content management system Joomla into a framework in order to reduce its development time. While this approach can dramatically reduce the development time, it makes the web application highly inflexible by placing the application into a CMS framework. Moreover, it relies heavily on Joomla plug-ins to provide functionality to the web application. Similarly, Ahlawat (2016) proposed to build a web application based on Wordpress. Though this approach can speed up the development process, it gains the speed by sacrificing its flexibility as it depends on the functionality provided by the CMS platform and its ecosystem.

**Plug-in Frameworks**

Alor-Hernández et al. (2015) proposed a new way of building web application based on the Adobe FLEX technology. This approach provides secure and reactive web applications as the application is pre-installed in the browser upon the first load. However, it requires clients to install a run-time environment on their systems before running the application. In addition, every time the runtime environment is required to be updated, the client needs to download and install it again manually. Similarly, a Silverlight framework is proposed by Appasami and Suresh (2009) which provides strong reactivity but also requires a run-time environment to run.

**Current Best Solution and Its Limitations**

Lamża et al. (2015) proposed a scalable and flexible web application framework seen in Fig. 2. This approach followed the MVC model by completely separating front end and back end. In this way, front end developers and back end developers can develop the web application at the same time as long as there are agreed Application Programming Interface (APIs) between them. This would significantly boost up the development speed. Furthermore, it enhances the model by moving the UI layer to a separate web server. In this way, the server that handles back end services is distinct from the one that handles front end UI layer. This makes the whole structure highly scalable. In addition, it applies Node.js as the UI layer server language to facilitate its development, which is echoed by other researchers (Cantelon et al., 2014). This is because Node.js application can be written completely in Javascript,
which has long been used as a front end language. This makes the UI layer easy to maintain as there is no steep learning curve required for front end developers.

On the back end, it breaks the monolithic back end layer into multiple individual web services and connects front end and back end using an API gateway. This dramatically reduces the complexity of maintaining the back end services. Moreover, since the back end server is individual, it can be developed using battle-tested languages and frameworks such as Java or C#. Therefore, it provides solid security to the framework.

Limitations

This framework excels at development speed, scalability and security. However, it is not optimal in terms of page reactiveness. The UI layer is currently separating from the back end server, which makes the development and maintenance easier. However, the UI layer still requires frequent communications with web browser in order for a web application to render properly. For example, each time users try to navigate through a web application by clicking links on it, the web browser would send requests to the web server that hosts the UI layer and then the server would respond to the web browser by sending back a combination of HTML, CSS and Javascript, or a pre-rendered UI view. In either case, frequent communications between web browser and web server is required in order for the web application to function. The web application would respond poorly if the network is unstable in this framework. Therefore, page reactiveness is hampered by these frequent communications and would lead to poor user experience. Though there are defects in how this framework handles its UI layer (Fig. 3), it seems that the back-end framework used in this approach can be used in combination of front-end router technologies to provide a better reactive experience. Further researches have to be done on this possibility of fusing the back-end framework with front-end router technologies.

Web applications have now become a central part of the internet. The need for making web applications more reactive is of top priority. This paper has presented reviews of recent web application frameworks. It is concluded that many researchers have been focusing on proposing frameworks that address issues in the back-end, such as improving the scalability and security of the framework. While some frameworks introduced technologies to enhance reactiveness, they can be further optimized and improved.

The proposed solution will be based on the solution proposed by Lamža et al. (2015) and will be focusing on enhancing the reactiveness of the web application in an effort to provide a better user experience and reduce server resource.

Proposed Model

The proposed framework in Fig. 4 is built upon the framework proposed by Lamža et al. (2015). It mainly addresses the page reactiveness issue by integrating two libraries into the UI layer, namely, React and Preact-router. By applying these libraries, it dramatically changes the way how web browser and web server communicates. In the proposed framework, all the HTML, CSS and Javascript files will be bundled before going to production stage. The module bundling, a single Javascript file including all the code will be served to the UI layer. By doing this, the whole web application HTML documents, CSS styles and Javascript files are completely loaded up in the first load. In addition, routing is also handled in the client web browser by Preact-router. Therefore, no request is sent when users navigate around a web application. This is because all the pages have been loaded upfront into the browsers. This eliminates the needs for sending requests from a web browser to a web server and waiting for its responses. As a result, the reactiveness of the web app would improve significantly.

The two key libraries that are added to the framework are Preact (Preact, n.d.) and Preact-router (n.d.). Preact provides an ultra-thin Virtual Document Object Model (DOM) on top of the normal DOM. By introducing a Virtual DOM, direct manipulation of DOM elements is reduced to a minimal extent. Any UI changes would be first recorded in the Virtual DOM and then compared with the DOM to make the minimal DOM element updates. Since manipulating DOM is a costly operation which takes a certain amount of time, minimizing the need to manipulate DOM is an effective way to increase the reactiveness of web applications especially in rich interface ones. Furthermore, the Preact library is lightweight itself, which accounts for only 10KB. This small size makes the loading time of the library negligible. Therefore, Preact speeds up the page reactiveness significantly by replacing the traditional DOM with Virtual DOM, while not affecting the page load time in a dramatic way.

Fig. 3: The UI Layer of the framework

---

DOI: 10.3844/ajassp.2017.1081.1092
Preact-router is a library that keeps web application UI in sync with Uniform Resource Locator (URL). This library makes it possible to handle routing in the front-end. Without Preact-router, each user click on a page would send a request to the web server for a new HTML document. This operation would break the reactiveness of a web application as user has to wait for the web server to respond as well as for the web browser to re-render the HTML document. By introducing the routing library, it reduces the needs to send requests for page contents from the web server when users navigating through web applications. Therefore, it remarkably improves the reactiveness of web applications as fewer communications between web client and web server are required.

One of the biggest gains in the proposed framework is the web application becomes more reactive. After the initial load, the whole web application is in the browser, so no other server request for HTML documents is made when user navigate through the web application later on. This increases the user experience dramatically because it provides a smoother interactions that is similar to client applications. Furthermore, by bundling all the HTML, CSS and Javascript in one file and injecting it into the browser, it reduces significantly the requests required to fetch all the necessary files. By doing this, it potentially saves a large amount of server resources as well as bandwidth.

However, one prominent disadvantage of the proposed solution is a web application built using this approach would take longer to load for the first time compared with other traditional web applications. This is because it tries to load up all the HTML, CSS and Javascript files in one go.

Test on the Proposed Framework

It is necessary to implement testing to make an unbiased determination on whether the proposed new framework performs better than the current best solution in terms of web application reactiveness. The final decision was made to test the two frameworks on a single local machine in order to control external factors that affect the test, such as inconsistent bandwidth and packet loss, which might have caused the testing results to be misrepresented.

Testing Environment Setup

All testing were performed on a macOS Sierra system with a version of 10.12.3, which utilized a 2.2 GHz Intel Core i7 processor with 16.0 GB of RAM.

Loading time of the web pages were measured using Lori (Life-of-request info) 0.2.0.20080521.1 and The Addon Bar 3.2.9-compat-fixed-4 extensions with Firefox 53.0.2 (64-bit) and Chrome DevTools with Google Chrome Version 58.0.3029.110 (64-bit).

Firefox Browser

The Lori Firefox extension is designed to monitor the length of time required to completely load a web page in Firefox browser (Lori, n.d.). It measures the following metrics:

- Time to First Byte (TTFB): How long it took to see the first byte from a remote server
- Time to Complete (TTC): How long it took to display the page
- Page size: Number of bytes used to display the page

Chrome Browser

The Chrome DevTools is used to measure the following metrics:

- DOMContentLoaded: How long it took to load a HTML document
- Load: How long it took to load a HTML document and its dependent resources
Test Case

In order to test the load times for both frameworks, UI layer of a web application composed of 4 static HTML pages, 16 style sheets, 20 script files and 10 images was created using the reviewed and proposed frameworks respectively. All the files for the web application are listed in the appendices Table from 3 to 6. Additionally, Fig. 6 to 12 and Table 7 and 8 from appendices illustrates more details on the files based on different file types.

Testing Procedure

Two testing websites seen in Fig. 10 from appendices were built with the structure listed in the testing case section using two framework solutions respectively. The two websites were run 10 times in cache-free Firefox and Chrome browsers to test their loading speed and reactivity. The test was started on the Home tag, then navigated in the order of Portfolio, Courses and Tutorials to complete a full circle. Finally, the results will be analyzed to determine which framework is more reactive.

Testing Results

The 10 test results from Firefox browser is listed in Table 1 a screenshot of the execution timeline is provided in appendices (Fig. 8).

| Test Case | In order to test the load times for both frameworks, UI layer of a web application composed of 4 static HTML pages, 16 style sheets, 20 script files and 10 images was created using the reviewed and proposed frameworks respectively. All the files for the web application are listed in the appendices Table from 3 to 6. Additionally, Fig. 6 to 12 and Table 7 and 8 from appendices illustrates more details on the files based on different file types. | Testing Procedure | Two testing websites seen in Fig. 10 from appendices were built with the structure listed in the testing case section using two framework solutions respectively. The two websites were run 10 times in cache-free Firefox and Chrome browsers to test their loading speed and reactivity. The test was started on the Home tag, then navigated in the order of Portfolio, Courses and Tutorials to complete a full circle. Finally, the results will be analyzed to determine which framework is more reactive. | Testing Results | The 10 test results from Firefox browser is listed in Table 1 a screenshot of the execution timeline is provided in appendices (Fig. 8). |
| Test No. | Page | Solution from Lamza, Marzec and Wrobel | Proposed Solution | Test No. | Page | Solution from Lamza, Marzec and Wrobel | Proposed Solution |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Home | 0.043 | 0.411 | 2.36 | 0.088 | 1.104 | 2.82 |
| Portfolio | 0.054 | 0.395 | 2.37 | 0.000 | 0.000 | 2.82 |
| Course | 0.043 | 0.369 | 2.36 | 0.000 | 0.000 | 2.82 |
| Tutorial | 0.048 | 0.390 | 2.36 | 0.000 | 0.000 | 2.82 |
| 2 | Home | 0.048 | 0.372 | 2.36 | 0.047 | 0.372 | 2.37 |
| Portfolio | 0.047 | 0.372 | 2.37 | 0.000 | 0.000 | 2.82 |
| Course | 0.049 | 0.373 | 2.36 | 0.000 | 0.000 | 2.82 |
| Tutorial | 0.046 | 0.381 | 2.36 | 0.000 | 0.000 | 2.82 |
| 3 | Home | 0.058 | 0.392 | 2.36 | 0.106 | 1.123 | 2.82 |
| Portfolio | 0.043 | 0.379 | 2.37 | 0.000 | 0.000 | 2.82 |
| Course | 0.043 | 0.381 | 2.36 | 0.000 | 0.000 | 2.82 |
| Tutorial | 0.043 | 0.383 | 2.36 | 0.000 | 0.000 | 2.82 |
| 4 | Home | 0.051 | 0.411 | 2.36 | 0.097 | 1.204 | 2.82 |
| Portfolio | 0.043 | 0.402 | 2.37 | 0.000 | 0.000 | 2.82 |
| Course | 0.035 | 0.378 | 2.36 | 0.000 | 0.000 | 2.82 |
| Tutorial | 0.051 | 0.394 | 2.36 | 0.000 | 0.000 | 2.82 |
| 5 | Home | 0.048 | 0.373 | 2.36 | 0.081 | 1.134 | 2.82 |
| Portfolio | 0.049 | 0.371 | 2.37 | 0.000 | 0.000 | 2.82 |
| Course | 0.050 | 0.356 | 2.36 | 0.000 | 0.000 | 2.82 |
| Tutorial | 0.041 | 0.371 | 2.36 | 0.000 | 0.000 | 2.82 |
| 6 | Home | 0.047 | 0.362 | 2.36 | 0.092 | 1.133 | 2.82 |
| Portfolio | 0.043 | 0.368 | 2.37 | 0.000 | 0.000 | 2.82 |
| Course | 0.048 | 0.373 | 2.36 | 0.000 | 0.000 | 2.82 |
| Tutorial | 0.049 | 0.382 | 2.36 | 0.000 | 0.000 | 2.82 |
| 7 | Home | 0.052 | 0.392 | 2.36 | 0.106 | 1.136 | 2.82 |
| Portfolio | 0.046 | 0.373 | 2.37 | 0.000 | 0.000 | 2.82 |
| Course | 0.048 | 0.382 | 2.36 | 0.000 | 0.000 | 2.82 |
| Tutorial | 0.049 | 0.381 | 2.36 | 0.000 | 0.000 | 2.82 |
| 8 | Home | 0.053 | 0.411 | 2.36 | 0.117 | 1.211 | 2.82 |
| Portfolio | 0.049 | 0.402 | 2.37 | 0.000 | 0.000 | 2.82 |
| Course | 0.039 | 0.378 | 2.36 | 0.000 | 0.000 | 2.82 |
| Tutorial | 0.053 | 0.394 | 2.36 | 0.000 | 0.000 | 2.82 |
| 9 | Home | 0.045 | 0.373 | 2.36 | 0.101 | 1.214 | 2.82 |
| Portfolio | 0.044 | 0.389 | 2.37 | 0.000 | 0.000 | 2.82 |
| Course | 0.054 | 0.353 | 2.36 | 0.000 | 0.000 | 2.82 |
| Tutorial | 0.045 | 0.371 | 2.36 | 0.000 | 0.000 | 2.82 |
| 10 | Home | 0.047 | 0.362 | 2.36 | 0.096 | 1.178 | 2.82 |
| Portfolio | 0.043 | 0.363 | 2.37 | 0.000 | 0.000 | 2.82 |
| Course | 0.044 | 0.371 | 2.36 | 0.000 | 0.000 | 2.82 |
| Tutorial | 0.051 | 0.362 | 2.36 | 0.000 | 0.000 | 2.82 |
Table 2: Test results in Chrome browser

| Test in Chrome | Solution from Lamza, Marzec and Wrobel | Proposed Solution |
|----------------|----------------------------------------|-------------------|
|                | DOM Content Loaded (ms) | Load (ms) | Size (MB) | DOM Content Loaded (ms) | Load (ms) | Size (MB) |
| 1 Home         | 465 | 478 | 2.36 | 685 | 721 | 2.82 |
| Portfolio      | 532 | 565 | 2.37 | 0 | 0 | 2.82 |
| Course         | 487 | 499 | 2.36 | 0 | 0 | 2.82 |
| Tutorial       | 475 | 487 | 2.36 | 0 | 0 | 2.82 |
| 2 Home         | 458 | 469 | 2.36 | 699 | 736 | 2.82 |
| Portfolio      | 511 | 531 | 2.37 | 0 | 0 | 2.82 |
| Course         | 479 | 480 | 2.36 | 0 | 0 | 2.82 |
| Tutorial       | 458 | 486 | 2.36 | 0 | 0 | 2.82 |
| 3 Home         | 457 | 468 | 2.36 | 720 | 755 | 2.82 |
| Portfolio      | 512 | 551 | 2.37 | 0 | 0 | 2.82 |
| Course         | 489 | 501 | 2.36 | 0 | 0 | 2.82 |
| Tutorial       | 463 | 476 | 2.36 | 0 | 0 | 2.82 |
| 4 Home         | 452 | 468 | 2.36 | 768 | 806 | 2.82 |
| Portfolio      | 531 | 514 | 2.37 | 0 | 0 | 2.82 |
| Course         | 451 | 468 | 2.36 | 0 | 0 | 2.82 |
| Tutorial       | 458 | 487 | 2.36 | 0 | 0 | 2.82 |
| 5 Home         | 459 | 508 | 2.36 | 798 | 838 | 2.82 |
| Portfolio      | 521 | 514 | 2.37 | 0 | 0 | 2.82 |
| Course         | 461 | 473 | 2.36 | 0 | 0 | 2.82 |
| Tutorial       | 475 | 498 | 2.36 | 0 | 0 | 2.82 |
| 6 Home         | 479 | 462 | 2.36 | 756 | 801 | 2.82 |
| Portfolio      | 526 | 589 | 2.37 | 0 | 0 | 2.82 |
| Course         | 465 | 564 | 2.36 | 0 | 0 | 2.82 |
| Tutorial       | 457 | 479 | 2.36 | 0 | 0 | 2.82 |
| 7 Home         | 451 | 446 | 2.36 | 773 | 815 | 2.82 |
| Portfolio      | 543 | 578 | 2.37 | 0 | 0 | 2.82 |
| Course         | 508 | 521 | 2.36 | 0 | 0 | 2.82 |
| Tutorial       | 487 | 501 | 2.36 | 0 | 0 | 2.82 |
| 8 Home         | 465 | 478 | 2.36 | 707 | 746 | 2.82 |
| Portfolio      | 514 | 533 | 2.37 | 0 | 0 | 2.82 |
| Course         | 478 | 499 | 2.36 | 0 | 0 | 2.82 |
| Tutorial       | 480 | 501 | 2.36 | 0 | 0 | 2.82 |
| 9 Home         | 489 | 502 | 2.36 | 743 | 779 | 2.82 |
| Portfolio      | 541 | 557 | 2.37 | 0 | 0 | 2.82 |
| Course         | 476 | 498 | 2.36 | 0 | 0 | 2.82 |
| Tutorial       | 468 | 489 | 2.36 | 0 | 0 | 2.82 |
| 10 Home        | 472 | 494 | 2.36 | 725 | 762 | 2.82 |
| Portfolio      | 545 | 578 | 2.37 | 0 | 0 | 2.82 |
| Course         | 487 | 508 | 2.37 | 0 | 0 | 2.82 |
| Tutorial       | 475 | 490 | 2.36 | 0 | 0 | 2.82 |

Fig. 5: A comparison of web content load time in Chrome and Firefox browsers
The 10 test results from Chrome browser is listed in Table 2 a screenshot of the execution timeline is provided in appendices (Fig. 9).

Figure 5 is a comparison of web content load time in Chrome and Firefox browsers.

Results and Discussion

According to the readings from the tests performed in Chrome, the proposed framework required approximately 60% less time to load the whole web application and its dependent resources.

This is calculated by using the total load time of the proposed solution divided by the total load time of the current best solution. To take the Test No.1 case as an example, the calculation is 721/(478+565+499+487)-100% ≈ -64.46%

It is noted that, in the framework used by Lamza, Marzec and Wrobel, each page requires a separate page load, which is in stark contrast to the proposed solution where only the home page is required a page load. This is because the whole application is loaded into the browser upon the first page load in the proposed framework. After loading the whole application, there is no need to send requests to the web server when users navigate to portfolio, course and tutorial pages. This can be seen in the Table 6 where the load time for portfolio, course and tutorial pages is zero. By cutting the load time to zero for these pages, it makes the web application ultra-reactive. In addition, the proposed framework only sends one request to fetch the whole web application while the other framework does so in 4 requests. This reduces the burden of the web server and frees up its resources as less requests are sent from the web clients.

In the Firefox results, the readings showed approximately 30% less time to display the whole web application.

This is calculated by using the total TTC of the proposed solution divided by the total TTC of the current best solution. To take the Test No.1 case as an example, the calculation is 1.104/(0.411+0.395+0.369+0.390)-100% ≈ -29.46%.

The proposed framework loads up the entire application in the home page so subsequent page visits do not require any page load-up. Furthermore, it only sends one request to fetch the whole web application while the other framework does so in 4 requests. Other observations from the testing results in Firefox browser coincide with the findings from the Chrome one that the proposed framework makes the web application more reactive and consumes less web server resources.

Conclusion

Researchers have proven that the page load time has a significant impact on retaining visitors to a web application. Therefore, it is the aim of this paper to find a solution to improve the reactiveness of web applications. The proposed framework excelled at the web application load time as a whole because it loads everything up in one request. This makes the entire application ultra-reactive by reducing subsequent page visit load time to zero. In addition, it immensely saves server resources by cutting web client requests in a significant amount.

However, the reactivity comes at a price of increasing the first load time notably. Further research into reducing the first load time is needed to optimize the framework. Additionally, the experimentation was limited to a local machine running macOS system and other systems are not included in this testing. Further, the tests only utilized a simple web application composed of a limited amount of files. This limited scope provided useful data for analysis, but it was not comprehensive. Further testing with a complete set of data need to occur in the future.

Acknowledgement

We are grateful to Angelika Maag from the CSU study center for proof reading and making corrections to this article. Without their support, it would have not been possible to submit this in the current form.

Funding Information

The authors should acknowledge the funders of this manuscript and provide all necessary funding information.

Author’s Contributions

Zhixong Xiao: Investigate issues and challenges web applications frameworks. Propose and implemented a new UI layer framework that makes web application more reactive. Zhixong has tested the proposed framework in multiple test cases against the current best solution.

Chandana Prasad Withana: Supervised/worked closely with Zhixong during the analysis, design and experiment phases.

Abeer Alsadoon: Worked on the setup of the experiments and gave important suggestions on design of experiments.
Amr Elchouemi: Give the final review and approval for the manuscript to be submitted.

Ethics

Authors should address any ethical issues that may arise after the publication of this manuscript.

References

Ahlawat, N., 2016. Build a WordPress web application using WAMP.
Alor-Hernández, G., V.Y. Rosales-Morales and L.O. Colombo-Mendoza, 2015. Frameworks, Methodologies and Tools for Developing Rich Internet Applications. 1st Edn., IGI Global, ISBN-10: 1466664371, pp: 366.
Anderson, S., 2017. How fast should a website load in 2017?
Appasami, G. and J.K. Suresh, 2009. Developing device independent Visual Components for web applications using Affine Vector graphics and silver light framework. Int. J. Comput. Electr. Eng., 1: 496-496.
Balasubramanee, V., C. Wimalasena, R. Singh and M. Pierce, 2013. Twitter bootstrap and AngularJS: Frontend frameworks to expedite science gateway development. Proceedings of the IEEE International Conference on Cluster Computing, Sept. 23-27, IEEE Xplore Press, Indianapolis, USA, pp: 1-1. DOI: 10.1109/CLUSTER.2013.6702640
Cantelon, M., M. Harter, T.J. Holowaychuk and N. Rajlich, 2014. Node.js in action. Manning Publications Co.
Chansuwath, W. and T. Senivongse, 2016. A model-driven development of web applications using AngularJS framework. Proceedings of the IEEE/ACIS 15th International Conference on Computer and Information Science, Jun. 26-29, IEEE Xplore Press, Okayama, pp: 1-6. DOI: 10.1109/ICIS.2016.7550838
Dooley, R., 2012. Don't let a slow website kill your bottom line.
Formack, L., 2017. Amazon's 'secret weapon': Understanding how website experience can influence shoppers.
Lardinois, F., 2016. Gmail now has more than 1B monthly active users.
Lori (Life-of-request info). (n.d.). Add-ons. https://addons.mozilla.org/en-US/firefox/addon/lori-life-of-request-info/.
Meenakshi, S., 2015. Ruby on rails - an agile developer's framework. Int. J. Comput. Applic., 112: 7-11.
MF, 2017. How the Web works. Mozilla Foundation.
Nations, D., 2016. Improve your understanding of web applications.
Nikolić, L., G. Milosavljević and I. Dejanović, 2016. Framework for Web application development based on Java technologies and AngularJS. Proceedings of the 6th International Conference on Information Society and Technology, (IST’ 16).
Panchal, H.B., 2016. A web application based on the MVC architecture using the spring framework (Order No. 10196389). ProQuest Dissertations and Theses Global, (185482284).
Pop, D.P. and A. Altar, 2014. Designing an MVC model for rapid web application development. Proc. Eng., 69: 1172-1179. DOI: 10.1016/j.proeng.2014.03.106
Preact (n.d.). https://preactjs.com/
Preact-router (n.d.). https://github.com/developit/preact-router.
Priefer, D., 2014. Model-driven development of content management systems based on Joomla. Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering, Sept. 15-19, ACM, Vasteras, Sweden, pp: 911-914. DOI: 10.1145/2642937.2653474
Privat, L., 2014. Google maps: 1 billion monthly users.
Rahman, A.A. and D.S. Chitra, 2015. A framework for ultra-responsive light weight web application using AngularJS. Proceedings of the Online International Conference on Green Engineering and Technologies, Nov. 27-27, IEEE Xplore Press, Coimbatore, pp: 1-4. DOI: 10.1109/GET.2015.7453857
Safronov, M. and J. Winesett, 2014. Web application development with Yii 2 and PHP. Packt Publishing Ltd.
Song, G., 2014. The reconstruction pattern of MVC. Int. J. u-e- Service Sci. Technol., 7: 147-156. DOI: 10.14257/ijunesst.2014.7.2.14
Vohra, D., 2014. JRuby Rails Web Application Development. 1st Edn., Springer International Publishing, Heidelberg, ISBN-10: 3319039342, pp: 67.
## Appendix A

### Table 3: All the HTML files

| No. | Static HTML page     | Size |
|-----|----------------------|------|
| 1   | index.html           | 2 KB |
| 2   | courses.html         | 1 KB |
| 3   | portfolio.html       | 4 KB |
| 4   | tutorials.html       | 1 KB |

### Table 4: All the CSS files

| No. | Style Sheet            | Size |
|-----|------------------------|------|
| 1   | style.css              | 5 KB |
| 2   | animate.min.css        | 53 KB|
| 3   | hint.min.css           | 10 KB|
| 4   | hover.min.css          | 98 KB|
| 5   | loaders.min.css        | 41 KB|
| 6   | balloon.min.css        | 5 KB |
| 7   | btn.min.css            | 33 KB|
| 8   | cshakemin.css          | 22 KB|
| 9   | flag-icon.min.css      | 33 KB|
| 10  | github-markdown.min.css| 12 KB|
| 11  | grid.min.css           | 5 KB |
| 12  | mobi.min.css           | 10 KB|
| 13  | sanitize.min.css       | 3 KB |
| 14  | spectre.min.css        | 42 KB|
| 15  | tuft.min.css           | 7 KB |
| 16  | zocial.min.css         | 45 KB|

### Table 5: All the Javascript files

| No. | Script                  | Size  |
|-----|-------------------------|-------|
| 1   | jquery-3.2.1.min.js     | 87 KB |
| 2   | moment.min.js           | 51 KB |
| 3   | bootstrap.min.js        | 37 KB |
| 4   | jquery-ui.min.js        | 254 KB|
| 5   | lodash.min.js           | 71 KB |
| 6   | beautify.min.js         | 33 KB |
| 7   | chroma.min.js           | 37 KB |
| 8   | Draft.min.js            | 128 KB|
| 9   | intercooler.min.js      | 30 KB |
| 10  | is.min.js               | 13 KB |
| 11  | js.cookie.min.js        | 2 KB  |
| 12  | jsoneditor.min.js       | 162 KB|
| 13  | jsplumb.min.js          | 198 KB|
| 14  | jstree.min.js           | 135 KB|
| 15  | matter.min.js           | 86 KB |
| 16  | mo.min.js               | 130 KB|
| 17  | offline.min.js          | 10 KB |
| 18  | p5.min.js               | 285 KB|
| 19  | sir-trevor.min.js       | 377 KB|
| 20  | vex.min.js              | 9 KB  |

### Table 6: All the Images files

| No. | Image       | Size |
|-----|-------------|------|
| 1   | test1.jpg   | 28 KB|
| 2   | test2.jpg   | 74 KB|
| 3   | test3.jpg   | 23 KB|
| 4   | test4.jpg   | 25 KB|
| 5   | test5.jpg   | 168 KB|
| 6   | test6.jpg   | 75 KB|
| 7   | test7.jpg   | 37 KB|
| 8   | test8.jpg   | 23 KB|
| 9   | test9.jpg   | 85 KB|
| 10  | test10.jpg  | 61 KB|

Fig. 6: Number of image objects across different rank ranges

Fig. 7: Number of CSS objects across different rank ranges

Fig. 8: Number of Javascript objects across different rank ranges
Fig. 9. Number of various file types

Fig. 10. Different file types contributed to total bytes

Fig. 11. Number of objects across different rank ranges

Fig. 12. The testing web application
**Table 7: Firefox browser execution timeline**

| Status | Method | File | Domain | Cause | Type | Transfer... | Size | 2 ms | 5 ms | 3 ms |
|--------|--------|------|--------|-------|------|-------------|------|------|------|------|
| 304    | GET    | jQuery-3.2.1.min.js | localhost:3000 | script | js   | 84.83 KB | 36.18 KB | 35.85 KB | 35.85 KB | 35.85 KB |
| 304    | GET    | moment.min.js | localhost:3000 | script | js   | 50.26 KB | 50.26 KB | 50.26 KB | 50.26 KB | 50.26 KB |
| 304    | GET    | bootstrap.min.js | localhost:3000 | script | js   | 36.18 KB | 36.18 KB | 36.18 KB | 36.18 KB | 36.18 KB |
| 304    | GET    | jQuery-1.11.0.js | localhost:3000 | script | js   | 241.72 KB | 241.72 KB | 241.72 KB | 241.72 KB | 241.72 KB |
| 304    | GET    | bootstrap.min.js | localhost:3000 | script | js   | 69.15 KB | 69.15 KB | 69.15 KB | 69.15 KB | 69.15 KB |
| 304    | GET    | Draft.min.js | localhost:3000 | script | js   | 72.05 KB | 72.05 KB | 72.05 KB | 72.05 KB | 72.05 KB |
| 304    | GET    | is.min.js | localhost:3000 | script | js   | 12.87 KB | 12.87 KB | 12.87 KB | 12.87 KB | 12.87 KB |
| 304    | GET    | cookies.min.js | localhost:3000 | script | js   | 1.70 KB | 1.70 KB | 1.70 KB | 1.70 KB | 1.70 KB |
| 304    | GET    | mo.min.js | localhost:3000 | script | js   | 127.39 KB | 127.39 KB | 127.39 KB | 127.39 KB | 127.39 KB |
| 304    | GET    | offline.min.js | localhost:3000 | script | js   | 9.42 KB | 9.42 KB | 9.42 KB | 9.42 KB | 9.42 KB |
| 304    | GET    | p5.min.js | localhost:3000 | script | js   | 277.91 KB | 277.91 KB | 277.91 KB | 277.91 KB | 277.91 KB |
| 304    | GET    | beauty.min.js | localhost:3000 | script | js   | 31.85 KB | 31.85 KB | 31.85 KB | 31.85 KB | 31.85 KB |
| 304    | GET    | crons.min.js | localhost:3000 | script | js   | 35.85 KB | 35.85 KB | 35.85 KB | 35.85 KB | 35.85 KB |
| 304    | GET    | treeview.min.js | localhost:3000 | script | js   | 29.22 KB | 29.22 KB | 29.22 KB | 29.22 KB | 29.22 KB |
| 304    | GET    | jshandler.min.js | localhost:3000 | script | js   | 158.04 KB | 158.04 KB | 158.04 KB | 158.04 KB | 158.04 KB |
| 304    | GET    | jspris.min.js | localhost:3000 | script | js   | 192.92 KB | 192.92 KB | 192.92 KB | 192.92 KB | 192.92 KB |
| 304    | GET    | jatree.min.js | localhost:3000 | script | js   | 132.23 KB | 132.23 KB | 132.23 KB | 132.23 KB | 132.23 KB |
| 304    | GET    | matter.min.js | localhost:3000 | script | js   | 84.04 KB | 84.04 KB | 84.04 KB | 84.04 KB | 84.04 KB |
| 304    | GET    | sir-treeview.min.js | localhost:3000 | script | js   | 369.02 KB | 369.02 KB | 369.02 KB | 369.02 KB | 369.02 KB |
| 304    | GET    | vex.min.js | localhost:3000 | script | js   | 8.71 KB | 8.71 KB | 8.71 KB | 8.71 KB | 8.71 KB |
| 304    | GET    | bundle.js | localhost:3000 | script | js   | 799.67 KB | 799.67 KB | 799.67 KB | 799.67 KB | 799.67 KB |
| 200    | GET    | test1.png | localhost:3000 | img | png | 1 B | 1 B | 1 B | 1 B | 1 B |
| 200    | GET    | test2.png | localhost:3000 | img | png | 1 B | 1 B | 1 B | 1 B | 1 B |
| 200    | GET    | images.png | localhost:3000 | img | png | 1 B | 1 B | 1 B | 1 B | 1 B |
| 200    | GET    | inline-1493547466657 | localhost:3000 | img | json | 79 B | 79 B | 79 B | 79 B | 79 B |
| 101    | GET    | websocket | localhost:3000 | websocket | plain | 0 B | 0 B | 0 B | 0 B | 0 B |
| 200    | GET    | site.css.map | localhost:3000 | css | json | 1.59 KB | 1.59 KB | 1.59 KB | 1.59 KB | 1.59 KB |
| 200    | GET    | mob.min.css.map | localhost:3000 | css | json | 1.59 KB | 1.59 KB | 1.59 KB | 1.59 KB | 1.59 KB |

**Table 8: Chrome browser execution**

![Chrome browser execution](image-url)