DEGREE PROJECT IN MEDIA TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2018

Analysis of web performance optimization and its impact on user experience

AH ZAU MARANG

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Analysis of web performance optimization and its impact on user experience

Ah Zau Marang
KTH Royal Institute of Technology
Stockholm, Sweden
marang@kth.se

Supervisor: Filip Kis
Examiner: Haibo Li

Analys av optimering av webbprestanda och dess inverkan på användarupplevelse

Ah Zau Marang
Kungliga Tekniska högskolan (KTH)
Stockholm, Sverige
marang@kth.se

Handledare: Filip Kis
Examinator: Haibo Li

ABSTRACT
User experience (UX) is one of the most popular subjects in the industry nowadays and plays a significant role in the business success. As the growth of a business depends on customers, it is essential to emphasize on the UX that can help to enhance customer satisfaction. There has been statements that the overall end-user experience is to a great extent influenced by page load time, and that UX is primarily associated with performance of applications. This paper analyzes the effectiveness of performance optimization techniques and their impact on user experience. Principally, the web performance optimization techniques used in this study were caching data, fewer HTTP requests, Web Workers and prioritizing content. A profiling method Manual Logging was utilized to measure performance improvements. A UX survey consists of User Experience Questionnaire (UEQ) and three qualitative questions, was conducted for UX testing before and after performance improvements. Quantitative and qualitative methods were used to analyze collected data. Implementations and experiments in this study are based on an existing tool, a web-based application. Evaluation results show an improvement of 45% on app load time, but no significant impact on the user experience after performance optimizations, which entails that web performance does not really matter for the user experience. Limitation of the performance techniques and other factors that influence the performance were found during the study.

ABSTRACT
Användarupplevelse (UX) är idag en av de mest populära ämnena inom IT-branschen och spelar en viktig roll i affärsframgångar. Eftersom tillväxten av ett företag beror på kunder är det viktigt att betona på UX som kan bidra till att öka kundnöjdheten. Det har konstaterats att den övergripande slutanvändarupplevelsen i stor utsträckning påverkas av sidladdningstiden och att UX huvudsakligen är förknippad med applikationernas prestanda. I denna studie analyseras effektiviteten av optimeringstekniker och deras inverkan på användarupplevelse. Huvudsakligen, optimeringstekniker som användes i denna studie var cache-lösning, färre HTTP-förfrågningar, Web Workers och prioritering av innehåll. Profileringsmetoden "Manual Logging" användes för att mäta prestandaförbättringar. En enkätutvärdering som består av User Experience Questionnaire (UEQ) och tre kvalitativa frågor, genomfördes med fokus på användarupplevelsen före och efter prestandaförbättringar. Kvantitativa och kvalitativa metoder användes för att analysera insamlade data. Implementeringar och experiment i denna studie är baserade på ett befintligt verktyg, en webbaserad applikation. Utvärderingsresultatet visar en förbättring på 45% när det gäller sidladdningstid men ingen signifikant inverkan på användarupplevelsen efter prestandaförbättringar, vilket innebär att webbprestanda inte spelar någon roll för användarupplevelsen. Begränsning av optimeringstekniker och andra faktorer som påverkar prestationen hittades under studien.

Author Keywords
Web performance, user experience (UX), Caching data, performance measurement

ACM Classification Keywords
• Human-centered computing~Human computer interaction (HCI)

• General and reference~Performance

• Information systems~Web applications

INTRODUCTION
During the last decade, user experience became a buzzword in the field of Human-Computer Interaction (HCI) [5] and is becoming a key term in the world of interactive product design [6]. The term itself lacks proper definition and is associated with many different meanings, ranging from traditional usability to beauty, hedonic, affective or experiential aspects of technology use. Thus, UX has become more important compared to manifestations of usability such as productivity or learnability [5].

Good user experience is increasingly demanded by customers and end-users. Thereby, UX is highly important for business because good UX improves software quality and increases economic value, whereas bad UX drives customers away and leads to a bad reputation [1]. According to a study by Hogan, user experience is primarily associated with performance of applications as she argued that web performance is user experience and a website's experience determines how often users visit the website. In addition, Hogan stated that page load time and a responsive feeling of the website are a large part of the user experience [7]. According to Forrester Consulting, users expect a page to load in two seconds and after three seconds up to 40% of users abandon the website [8].

An example on how bad performance negatively impacts users is the situation that FPT development team encountered. FPT (Feature & Product Tracker) is a web-based application and used for project follow-up and planning, and accessed daily by users all around the world. An issue that users of FPT are suffering with is that the app is extremely slow as it processes and renders large amounts of data. FPT development team observed that this long load time and unresponsive interaction with the tool leads to a detrimental user experience and discourages usage of the tool. As a user of FPT said via the in-tool feedback: "The

FPT looks nice and have good features, but is more or less useless due to the extremely slow response to queries", the performance influences highly the user experience and the utility of the tool. Moreover, this issue also leads to making people spend time on waiting and waste money.

Web performance can be improved in many different ways. Some performance optimization techniques mentioned by Gardner [3] were to reduce the HTTP requests a page makes to the server, reduce page size and use caching. Another technique is Gzip which is a compression technique and used for reducing the size of files and data sent over the Internet [2]. Queries to databases can also cause slow web performance. Researchers have shown that using index structures can improve the performance in querying large databases [9].

Since there has been arguments that user experience is associated with a wide range of factors and not just performance, this paper focuses on improving web performance by implementing performance optimization techniques and evaluating their impact on user experience in order to inspect whether or not user experience is primarily associated with performance of applications. Implementations and experiments which are presented in this thesis are based on the existing tool, FPT. The outcome of this research is valuable for UX designers and developers in the field of HCI. The scope of this thesis does not include finding the best solution for all web applications. Measurement of performance is focused on app load time and initial view load time of all pages, and view load time when navigating to other pages from home. Furthermore, the performance measurement does not involve measuring browsers' speed and their JavaScript engine.

This paper makes the following contributions:

- Performance techniques analyzed: caching data, fewer HTTP requests, redesigning data flow in the app, improving rendering views, prioritizing content and working with web workers.
- User experience is tested using User Experience Questionnaire (UEQ) and three UX-related qualitative questions before and after performance improvements.
- The impact of performance optimization on UX is evaluated.
- Limitation of the performance techniques is discussed.

The rest of this paper is organized as following. Section 2 provides theory and related research. Section 3 provides method description. Section 4 provides performance improvement implementation. Section 5 provides evaluation results. Section 6 provides discussion, and the study is concluded in section 7.

THEORY AND RELATED RESEARCH

Slow page load time can be frustrating and annoying to users, and should be kept below 4 seconds [10]. In a study in 2004, a group of researchers evaluated website delays and how tolerant users are in the context of websites. In their study, they examined delay times of 0, 2, 4, 6, 8, 10, and 12 seconds in an experiment where 196 undergraduate students were asked to complete 9 search tasks, exploring a familiar and an unfamiliar site. Their outcomes indicate that long response time causes a noticeable decrease in performance, attitude and behavioral intention of a user. Therefore, page load time should not be greater than 8 seconds to promote a positive attitude of the user and should be kept below 4 seconds to encourage users to complete their task and visit the site again [10].

Prioritizing web content so that pages are shown to users sooner with the important content, is another approach to improve web performance. A research [11] that explores the approach of dynamically re-prioritizing web content, shows positive impact on user experience. In the study, a system KLOTSKI was developed to prioritize the most relevant content to a user's preferences. KLOTSKI consists of two parts: a back-end measurement engine that captures key invariant characteristics of a web page and a front-end proxy that uses the characteristics along with user preferences and client conditions to prioritize content. In development of KLOTSKI, three key challenges were addressed: (1) accounting for inter-resource dependencies on a page; (2) enabling fast selection and load time estimation for the subset of resources to be prioritized; and (3) developing a practical implementation that requires no changes to websites. Evaluation results of the study show that resources delivered within 2 seconds were increased from 25% to 60%.

Measuring performance of a web application has been a big challenge to developers due to how the performance of different browsers varies when processing Javascript [12]. Nägele et al. [12] have done a study concerning performance measurement methods at client-side. In their study, they evaluated several profiling approaches. A quantitative evaluation method was used to compare the different methods. To determine suitable methods, they made use of four properties (accuracy, impact, usability and portability) which were translated into four metrics. These metrics were then for each profiling method combined into one grade which expresses the suitability of each method for performance measurement at client-side. Out of all the evaluated methods, Manual Logging was found to be a considerable method for measuring performance of web applications. Manual Logging is a profiling method that involves adding statements to function declarations to create a profile tree at run-time which holds information about every executed function, function name, location and execution time [12]. This method is used in this study to

measure load times and evaluate the improvement of performance.

The purpose of caching data at client side is to reduce page latency and bandwidth usage [13]. There is a variety of data types that are cached at client side, e.g. pages with changing content [13], synchronized data [14], streaming media content [15], etc. A cache system can vary depending on the data type. In case of pages with changing content, a cache system can be implemented in the way that a page is stored in a user's browser cache and when the user requests the page the version of the page is controlled by a server. If the latest version of the page is in the browser cache, the page is retrieved from the cache otherwise from the server. The version of the page is checked in the way that a script is downloaded on page request that appends a previously established cookie value to the requested URL and triggers re-request of the URL. The server receives the cookie value and computes it based on the last modified time of the data used to generate the page [13].

Web Workers API is a mechanism that allows executing JavaScript applications on multiple threads [17]. It is intended to be used for keeping pages responsive on web browsers and improving user experience [18]. In the research of Okamoto et al. [18], a load distribution technique for a web server was proposed that utilizes Web Workers to distribute the calculations of the server to its clients. Motivation behind the technique was that the workload on the server (responsible for manipulating the behavior of game characters) for a real-time web application was heavy as more users logged in. As a result, the technique was found to be decreasing the CPU usage and increasing the communication load of the server. However, integrating the technique with a cache mechanism was believed to be a great combination as it can reduce the frequent database access and the data transmission, and in turn the CPU usage and the communication load [18].

METHOD DESCRIPTION

In order to inspect users' opinions on the FPT tool, UX testing was performed twice with a 2-weeks interval between: first before performance improvements and then after improvements. A survey consists of User Experience Questionnaire (UEQ), Appendix B, and three qualitative questions:

- What is your impression of the responsiveness in FPT (when navigating between views, filtering and interacting with the tool)?
- Which improvements of FPT would you value the most?
- Any additional comments or feedback?

was used for UX testing.

The original German version of UEQ was created in 2005 and consisted of 26 semantic differential items. Each item is presented by two terms with opposite meanings and scaled from -3 to +3: -3 represents the most negative answer (horribly bad UX), 0 a neutral answer and +3 the most positive answer (extremely good UX). The order of the terms is randomized, i.e. half of the items start with a positive term and the other half start with a negative term. The UEQ contains six scales:

- Attractiveness: Overall impression of the product. Do users like or dislike the product?
- Efficiency: Can users solve their tasks without unnecessary effort?

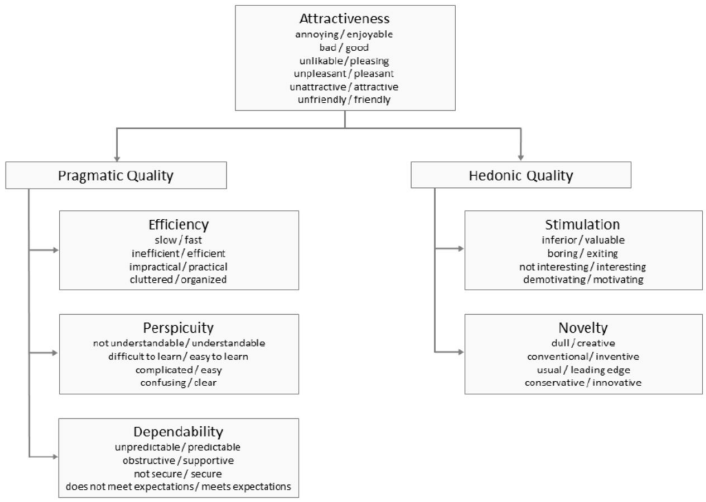


Figure 1: Assumed scale structure of the UEQ. Adapted from Schrepp, Martin. (2015). User Experience Questionnaire Handbook. 10.13140/RG.2.1.2815.0245. Copyright 2015 by Schrepp, Martin.

- Perspicuity: Is it easy to get familiar with the product? Is it easy to learn how to use the product?
- Dependability: Does the user feel in control of the interaction?
- Stimulation: Is it exciting and motivating to use the product?
- Novelty: Is the product innovative and creative? Does the product catch the interest of users? [1]

Each scale consists of 6 or 4 items as seen in Figure 1. The UEQ evaluation of two versions of a product is compared on the basis of the scale means for each UEQ scale. Values of the items are converted manually so that all items start from negative to positive when scale means are calculated.

The survey is conducted in the FPT tool itself in the way that it is prompted to users with 90 percent possibility whenever the users visit the website. For each UX testing, a user can solely answer to the survey once. The results from the survey were analyzed by using qualitative and quantitative data analysis to evaluate the impact of performance improvements on end-users.

Performance techniques: caching data, fewer HTTP requests, redesigning data flow in the app, improving rendering time of specific views, fetching only updated data from the server and working with web workers, were analyzed for performance improvements at client-side. Prioritizing important content is one approach that is used in this study in order to decrease rendering time and to improve swapping pages. Caching data is implemented using indexedDB which is a transactional local database embedded in the browser — a mechanism for persistently storing data in form of JSON objects collections inside a user's browser [16].

The performance was measured using the profiling method Manual Logging. Manual Logging is a method that records execution time of functions [4]. App load time and initial view load time were measured for all pages in FPT. App load time indicates the time it takes to load a page whereas the initial view load time indicates the time it takes to render a main view on page load. View load time for each navigation to other pages from home was measured as well. These load times were logged to the console in the web browser Chrome and recorded manually. The load times were measured 5 times for each page load and navigation, and median values of them were used to calculate average load times. The metrics were analyzed by using Statistical Data Analysis and average values were computed to figure out the improvement of performance.

Quantitative and qualitative data were collected from the survey for each UX testing. The analysis of answers on qualitative questions was conducted in the way that the answers were classified into different categories: negative, positive, load time, features and others, and thereby summarized for each category. Responses on the User Experience Questionnaire (UEQ) were evaluated by classifying them into the UEQ scales and scale means were calculated for each UEQ scale.

PERFORMANCE IMPROVEMENT IMPLEMENTATION

During analysis of all pages, three main views that remarkably take a long time to be loaded were found to be rendered 2-3 times on page load or view switch. As a part of performance improvement, this issue was addressed before implementing any performance technique.

Caching data was successfully implemented in FPT. The cache system is implemented in the way that 7 different datasets that were heavy to retrieve from the server are stored with a timestamp respectively in the IndexedDB. When a user loads a page, the data from IndexedDB are displayed first and new items are retrieved with the timestamp from the server. When the new items have arrived, it updates the items and timestamp in IndexedDB, and the displayed items in view. The design of cache system is demonstrated in Figure 2.

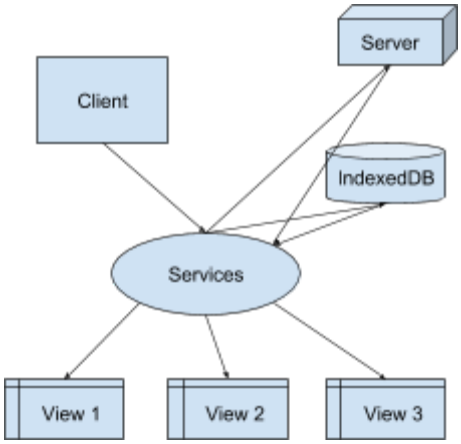


Figure 2: Cache system implemented in FPT

A successful technique that was effective to reduce rendering time of Backlog Item List view was incrementally adding rows to table and avoiding sorting rows during adding them into the table. Backlog Item List is one of the main views in FPT that is mostly used by users and usually takes more than 1 minute to be rendered with around 13000 items on page load, and contains a jQuery DataTable with

[1] User Experience Questionnaire Handbook (PDF Download Available). Available from: https://www.researchgate.net/publication/281973617_User_Experience_Questionnaire_Handbook [accessed May 09 2018].

140 columns. If no filters are applied, a dataset of around 13000 items (one row per item) are on page load added into the table which displays 50 rows per page. This view can sometimes, when internet connection is slow, cause timeout errors and make the page impossible to bring up. The solution to solve this issue was to add 50 rows first and display them to users and afterwards add 500 rows into the table in every 10 ms. This incrementally inserting rows into the table reduced the app load time of Backlog Item List by 84% (from 69634 ms to 11044 ms).

Load time of one HTTP request that retrieves, during the initial load of a page, a dataset (consists of around 30000 items) from the server and takes more than 1 minute, was managed and reduced to around 100ms. Instead of retrieving all the items, approximately 98% of the items were able to retrieve from another dataset at client-side. Unfortunately, reducing HTTP requests was impossible to do in short time because it involves restructuring of API which requires a lot of work at both server-side and client-side since the system is huge.

An additional issue with the tool was that pages did not get switched directly when clicked on menu-items in menu bar. Investigation of the problem gave two reasons why switching pages was unresponsive. One reason was that when the URL changes and the app loads a new main view component, data are fetched from a view service which subscribes to many other services and do filtering on a large number of items. This, in turn, makes Javascript do a lot of work and lock up the UI so that it does not have time to switch page and render the view until the fetching data is done. This issue was addressed by delaying fetching data from the view service on zero millisecond so that the JavaScript switches page and render the view first before it does a lot of work fetching data and lock up the UI. Some views of FPT are built in Backbone and embedded in Angular views, and they are displayed in a separate container. Backbone and Angular are front-end web application frameworks for building web, mobile web and native mobile applications. The directive "[hidden]" of Angular was used to toggle the container for Backbone views. Once a Backbone view is open, it freezes the UI for a moment when navigating to other pages. This depends on that it takes time for a large number of HTML elements in the container to be hidden by the "[hidden]" directive. Likewise, when navigating back to a Backbone view the obsolete HTML elements in the Backbone container were first displayed and removed when a new view is rendered, which made JavaScript do a lot of work and locked up the UI. This issue was addressed by programmatically removing all elements in the Backbone view container when navigating to other pages, instead of hiding them.

RESULTS

The results show that performance improvements had no impact on the overall user experience.

UX test before performance improvements

During 2 weeks of the survey period before performance improvements, 32 responses from end-users were collected. Evaluation of the responses on UEQ showed that FPT has quite good user experience before performance improvements were integrated in the system. The calculation of the UEQ scale means resulted in the value of around +1 for each UEQ scale. The scale means are demonstrated in Figure 3. The result shows that overall the users were positive, but we can notice a slightly lower score for novelty concerning the innovation and creativity of the product.

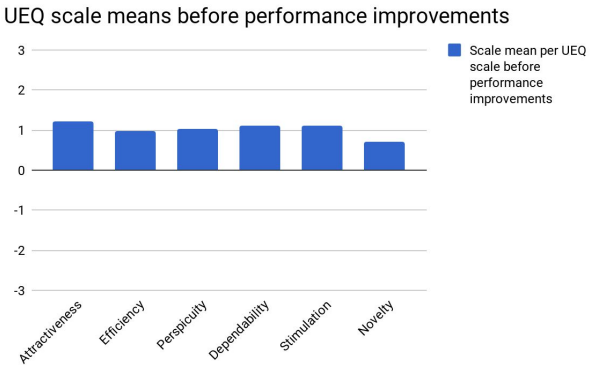


Figure 3: UEQ scale means of 32 responses on the User Experience Questionnaire (UEQ)

The qualitative analysis of the responses on the qualitative questions from the UX testing before performance improvements entailed that a variety of opinions was expressed by the end-users on the tool. Regarding the responsiveness of FPT when navigating between views, filtering and interacting with the tool, users expressed that the tool was slow in various ways. For instance, generating a view, loading UI elements and initial loading time without filters took long time as well as loading search results. Moreover, a user pointed out that the Backlog Item List view (page 11 in Figure 4) was impossible to bring up if no filters were selected. This unresponsiveness caused uncomfortable moments during meetings, and a strategy used to avoiding it was to preload the views before meeting. The slow response time was experienced as annoying as well. However, some users expressed positive views regarding user interface (UI), search, layout, filtering, usability, and features, whereas a few users experienced quick response time and filtering. In order to keep away from experiencing slow response time, some users avoid interacting with the tool, and instead use the same view and

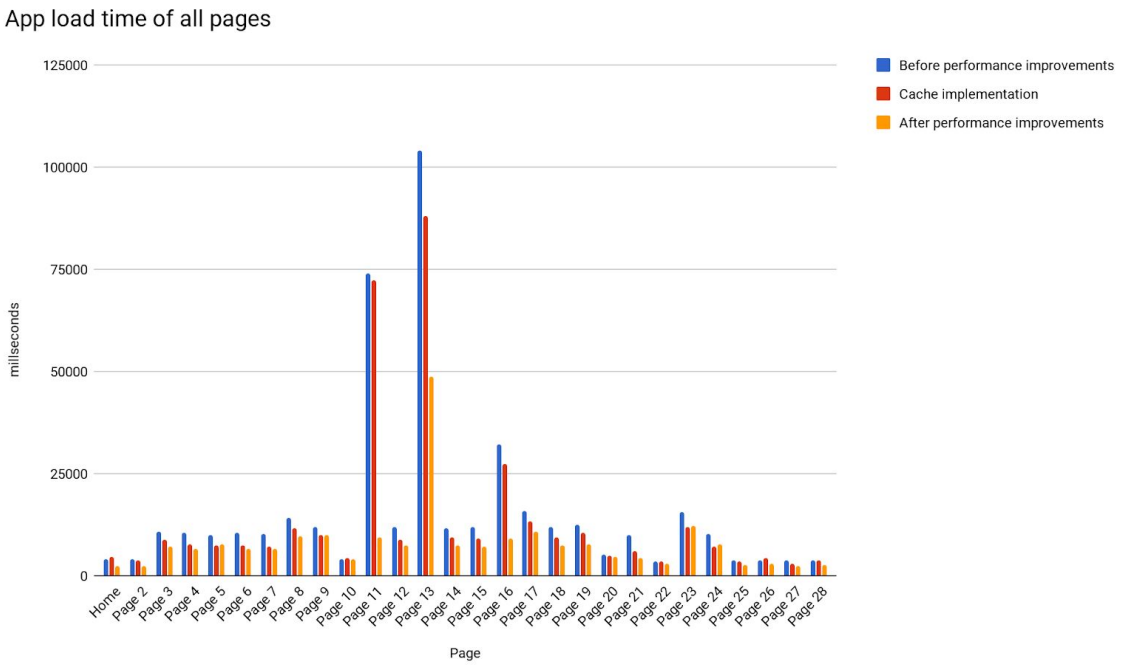


Figure 4: Median app load times of all pages of FPT on test server before/after performance improvements and with just implementation of caching data

filter every time when they visit FPT. A few users seem to be not really satisfied with the data quality as well.

Furthermore, users expressed that the improvements of FPT they would value the most were faster load time and better responsiveness such as page load time, rendering view and improvement of displaying search results. Other improvements requested by users were relating features and functionalities, such as ability to save specific features and "personal" data, possibility to save predefined queries/set of data, subscription on items, possibility to search in views, ability to track features within a specific area and a mechanism to flag "stale" data.

Other requested improvements from end-users were to improve the data quality and decrease visible fields in views. Columns with empty data seem to be annoying to users as it makes them do a lot of panning in order to see expected data, and users experienced that data in FPT were not always kept up to date. However, users seem to be very satisfied with quick response on their bug reports or improvement requests, and happy with the features that FPT offers.

Performance Improvements

According to the measurements of FPT load time on test server, the result shows a great improvement of load time for all loading types: app load time, initial view load time, and view load time for navigation to other pages from home. After all performance improvements were implemented in FPT, the app load time was found to be improved by 45%, whereas the initial view load time was improved by 53% and the view load time when navigating to other pages by 61%. Surprisingly, the implementation of caching data had less effect on load time than expected, namely the app load time was improved by only 18% and the initial view load time by 21%. Furthermore, the view load time when navigated to other pages from home was deteriorated by 2%. The metrics are presented in milliseconds in Figure 5.

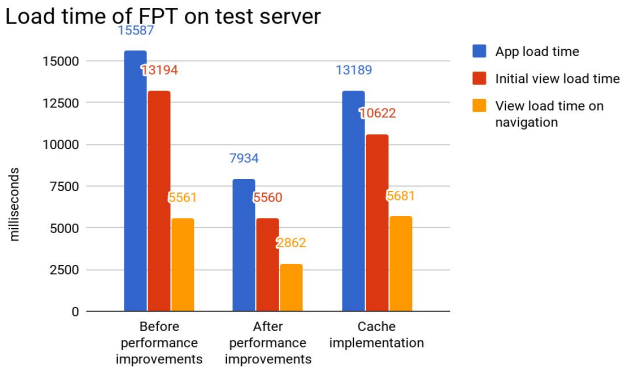


Figure 5: Average load times of FPT on test server before/after performance improvements and with just implementation of caching data.

An overview of median app load time for each page is presented in Figure 4. As can be seen in the graph, we can see that performance optimization has decreased the load time of all pages, and load time of page 11 and 16 has gone down significantly. If we also look at the graphs for initial view load time and load time on navigation, which are embedded in Appendix A in order to reduce the size of body text, there are four extremely slow pages and three of them have been improved exceptionally. However, there seem to be one page left that is still extremely slow for all load types.

UX test after performance optimizations

During the second survey period on the new version of the tool which performance improvements are integrated, 26 responses from users were recorded. Evaluation of the responses on UEQ showed a slight increase of overall user experience, namely concerning the attractiveness, efficiency, perspicuity and dependability. On the contrary, the result shows a tad decrease relating stimulation and a remarkable reduction of novelty. The comparison of UEQ scales before and after performance optimization can be seen in Figure 6.

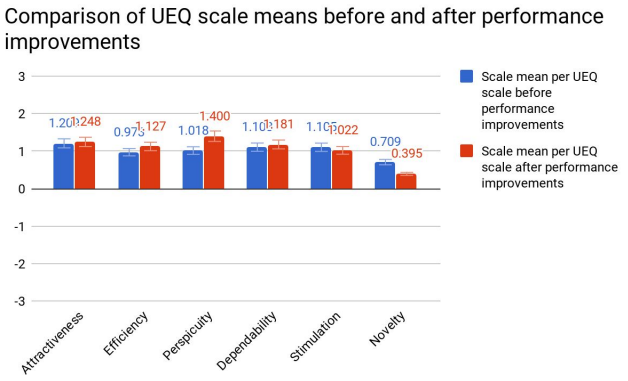


Figure 6: Comparison of UEQ scale means before and after performance improvements

The qualitative analysis of the responses on the qualitative questions from the second UX testing shows that the majority of the users who responded to the survey expressed positive impression towards responsiveness of the app regarding navigation between pages, filtering and interacting with the tool. In contrast, some users expressed dissatisfaction over the content in the app due to data inconsistency, difficulty finding data and unreliable search results. Relating to the question for further improvement of the app, faster load time is still a popular demand. Furthermore, the improvement of data quality is another strong request made by the users.

DISCUSSION

The evaluation result of the UEQ shows that there was no significant impact on UX by performance optimizations. As can be seen in Figure 6, four out of six UEQ scales got improved, but it shows very slight improvements in two of them: attractiveness and dependability. Furthermore, two other scales became slightly deteriorated.

If we perform a t-test using "t test calculator - QuickCalcs" of the service GraphPad[2] and compare the means of UEQ items from UX test 1 and UX test 2, Appendix D, the test result shows a p-value of 0.7339 (73%) which is greater than the traditional significance level 5%, and that a significant difference does not exist. If we also look at the means of UEQ items from both UX tests in Table 1, we can notice that there is no a significant difference between the means. Based on these results, we can conclude that the performance optimization does not really matter for UX. Thus, the aspect of Hogan [7] that page load time and a responsive feeling of a website are a large part of the user experience ought to be reconsidered.

	UX test 1	UX test 2
Mean of UEQ items	1.034578147	1.075675676

Table 1: Means of UEQ items from UX test 1 and UX test 2

The efficiency of the UEQ from first UX test resulted in around +1 which is a positive impression in terms of load time. However, there were many users desperately complaining about the load time. If we look at the average app load time of FPT before performance improvements in Figure 5, we can perceive that users have been enduring long page load time with an average app load time of 15-16 seconds. This denotes that the limit of 8 seconds pointed by Galletta et al. [10] is considerable. If we also have a look at the graphs in Figure 4 and Appendix A, we can notice that load times were around 8 seconds for most of the pages which might be the reason why the positive UEQ value was received as Galletta et al. [10] argued that a positive attitude of the user can be promoted by having page load time under 8 seconds. The reason why there were users desperately complaining about load time might be due to that the users open a heavy page every time when they visit FPT and that some pages which presumably are mostly used by users are the pages that take the longest time.

As can be seen in Figure 6, the scale mean for novelty has gone down significantly after performance optimizations. The reason behind this might be that users were experiencing a tragedy right before the second survey period. To be specific, there was a serious issue that data were not available for Firefox users at all after performance

[2] https://www.graphpad.com/quickcalcs/ttest1/?Format=C

improvements were integrated in the system. This was extremely annoying as a user said:

"Disappointing that FPT was out of service for so long for Firefox users. This fault was to obvious to pass even basic testing.....".

Likewise, the users were encountering data inconsistency, as a user mentioned in an answer to the survey:

"more accurate data, a lot of data is missing".

There were some reports regarding data inconsistency via mail as well. It is exceptionally interesting that user experience is to a great extent sensitive to data quality. These issues might be the cause to that the result does not show a significant impact on UX. Probably, the result may be different if there were no issues with the tool right before survey periods. In addition, after reading the comment in a response to the survey,

"Sometimes a bit slow, BUT worth waiting for (content beats speed)"

it might be interesting to find out how tolerant users are to enduring the slow load time in consideration of great data quality and how much the content can influence user experience. This could be a great topic for future research. Since there were many requests on features and functionalities from end users, it would be interesting to find out how much UX is sensitive to features and functionalities as well.

When an app is becoming large and there are functions listening to URL query parameters which are used to filter contents or to show search results, developers should exercise extreme caution when rendering a view with large data. If different contents of a view are displayed to end users depending on the query parameters, it is easy to miss that the content of the view changes programmatically several times on page load before anything is shown to a user. Thereby, the iterative rendering of a view is hard to be discovered without using breakpoints in the browser developer tools or logging something to the console. This was one factor that impaired the performance of some pages in FPT.

There was also a limitation of the performance technique "Caching data" at client-side. The weakness of caching data is that it causes more workloads at client-side because when new items are arrived from the server an algorithm is executed and it goes through the dataset fetched from cache, and adds/updates the items or removes the items that ought to be removed from the dataset. Thereafter, the dataset stored in cache is updated with the new dataset and the displayed items in views are updated with the new items as well. Certain views of FPT are implemented in the way that live-update is possible which means that any item can be updated without re-rendering the view. On the contrary, Backbone views of FPT requires re-rendering in order to update an item in a view and the whole new dataset need to be sent to the view again. This process is done for each dataset that is cached, which means that more datasets are stored in IndexedDB the more works need to be done at client-side on page load.

The most interesting performance technique "Web Workers" was unfortunately ineffective in FPT due to many reasons. Some of the reasons were that a class could not be sent to web workers, and web workers could not return an instantiated object with functions and libraries, as well as Angular services were unable to be imported into web workers. The web worker service "angular2-web-worker" was tested to do filtering of items (each item is an object that contains functions and properties.). Mainly, merging many different datasets and filtering of the merged items are the heavy work in FPT and not the calculations so the web worker was ineffective in FPT.

Manual logging method was not useful for a couple of pages because there were methods that return values as observable objects which means that it was demanding to measure the execution time of the methods. Thus, for some pages the app load time and view load time were measured by listening load event of an icon, image or an HTML element. For instance, when an image on home page was loaded, the app load timer and view load timer for the home page were stopped and logged into the console. Likewise, when the load event of an element in a table was triggered, the timers were stopped.

Computation of margin of error was performed for the metrics of the versions (with implementation of caching data and before any performance optimization), specifically the view load times when navigated to other pages from home. The margin of error for a 90% confidence level was computed using the following formula:

Margin of error = Critical value * Standard error of the statistic

$$MOE = z * \sigma/\sqrt{n}$$

MOE = margin of error

z = z-score (critical value)

σ = standard deviation

n = sample size

The margin of error for each navigation was calculated with the z-score of 1.645 for 90% confidence level, and the

average margin of error was computed for the two versions. Dividing the average margin of error by average view load time (load time when navigated to other pages from home) resulted in an average margin of error of 12.5% for the version with caching data and 7.9% for the version before any performance optimization. This indicates that the view load time when navigated to other pages from home being deteriorated with 2% by the implementation of caching data, can be an erroneous result as the margins of error are higher. The average margins of error are presented in Table 2. In addition, the optimal result ought to be that the implementation of caching data does not influence the view load time when navigated to other pages from home.

Navigate to other pages from Home	Average MOE (ms)	Average MOE (%)	Average view load time (ms)
Before performance improvements	438.20	7.88%	5560.85
Cache implementation	710.67	12.51%	5680.93

Table 2: Average margin of errors for view load time on navigation to other pages from home

If it is possible to conduct a survey for each performance technique, it will be interesting to see the improvement of UX for each performance technique. Doing UX testing for several performance improvements would allow one to have a better look at the coherence of web performance and UX. This will also help to identify the coherence bottlenecks of UX and web performance, which might be a good research in the future.

CONCLUSION

After implementation of performance improvements, app load time of FPT was found to be improved by 45%, whereas initial view load time was improved by 53% and view load time on navigation to other pages from home page by 61%. The implementation of caching data resulted in an improvement of 18% for app load time and 21% for initial view load time. It barely has an effect on view load time when navigating between pages within the app.

The comparison of UEQ scale means shows that there was no significant impact on the user experience after performance optimizations, which entails that web performance does not really matter for the user experience. Thus, the aspect of Hogan [7], that web performance is user experience and that page load time and a responsive feeling of a website are a large part of the user experience, ought to be reconsidered. Since load time for most pages was found to be around 8 seconds and the evaluation result shows positive UEQ values, the statement of Galletta et al. [10] that response time under 8 seconds increases attitude and behavioral intention of users, is considerable.

Limitations of the performance techniques were found during the study. For instance, to reduce fewer HTTP requests was not easy as it sounds because it involves restructuring of API which requires a lot of work to be done at database layer, client-side and server-side. Caching data causes a lot of workloads at client-side and therefore is not suitable to cache many datasets at the client-side. Web Workers was found to be not beneficial for applications that process a lot of objects which contain functions or methods.

ACKNOWLEDGMENTS

First of all, I would like to thank Meepo and Ericsson for giving me the opportunity to do this fascinating research. Special thanks to Sebastian Geidenstam and Henrik Ygge for your excellent project ideas, enlightenment and management throughout the project. I would like to express my gratitude to my supervisor at Meepo, Anders Hård, for his expert guidance and wisdom in front-end development that helped me understand a complex system at client-side in short time. I would like to extend my deep appreciation and thank Joacim Olofsson Eriksson for his effective strategies and solutions that helped me get through obstacles I encountered during the process. It is a pleasure to express my gratitude to Erick Chavez Alcarraz for sharing his outstanding knowledge of back-end development that assisted me in building APIs for survey and cache implementation. I also would like to express my gratitude to skillful UX designer, Christian Sterngren, for his expertise in UI design and animations that helped me achieve an awesome survey and better responsive feeling of the tool. It is an honor to express my gratitude to Henrik Ygge for his assistance in implementing cache system. Finally, I would like to thank my teachers and professors at KTH for their collaboration and encouragement, as well as classmates who provided helpful comments on the project and report. And thanks to everyone who took their precious time to participate in the survey.

REFERENCES

[1] Kuusinen, K., & Väänänen-Vainio-Mattila, K. (2012, October). How to make agile UX work more efficient: management and sales perspectives. In Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design (pp. 139-148). ACM.

[2] Souders, S. (2008). High-performance web sites. Communications of the ACM, 51(12), 36-41.

[3] Gardner, B. S. (2011). Responsive web design: Enriching the user experience. Sigma Journal: Inside the Digital Ecosystem, 11(1), 13-19.

[4] Nägele, T., Hooman, J., Zigterman, R., & Brul, M. (2015). Client-side performance profiling of JavaScript for web applications (Doctoral

dissertation, Master's thesis. Radboud University Nijmegen, the Netherlands).

[5] Hassenzahl, M., & Tractinsky, N. (2006). User experience-a research agenda. Behaviour & information technology, 25(2), 91-97.

[6] Battarbee, K., & Koskinen, I. (2005). Co-experience: user experience as interaction. CoDesign, 1(1), 5-18.

[7] Hogan, L. C. (2014). Designing for performance: weighing aesthetics and speed. " O'Reilly Media, Inc.".

[8] Forrester Consulting. (2009, August). eCommerce Web Site Performance Today: An Updated Look At Consumer Reaction To A Poor Online Shopping Experience. A commissioned study conducted on behalf of Akamai Technologies, Inc.

[9] Böhm, C., Berchtold, S., & Keim, D. A. (2001). Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. ACM Computing Surveys (CSUR), 33(3), 322-373.

[10] Galletta, D. F., Henry, R., McCoy, S., & Polak, P. (2004). Web site delays: How tolerant are users?. Journal of the Association for Information Systems, 5(1), 1.

[11] Butkiewicz, M., Wang, D., Wu, Z., Madhyastha, H. V., & Sekar, V. (2015, May). Klotski: Reprioritizing Web Content to Improve User Experience on Mobile Devices. In NSDI (Vol. 1, No. 1, pp. 2-3).

[12] Nägele, T., Hooman, J., Zigterman, R., & Brul, M. (2015). Client-side performance profiling of JavaScript for web applications (Doctoral dissertation, Master's thesis. Radboud University Nijmegen, the Netherlands).

[13] Chess, B., Goldberg, E., & Bailey, W. E. (2011). U.S. Patent No. 7,970,816. Washington, DC: U.S. Patent and Trademark Office.

[14] Nair, N. K., Nirmal, D., Turaga, S. C., & Wisneski, D. J. (2014). U.S. Patent No. 8,725,679. Washington, DC: U.S. Patent and Trademark Office.

[15] Klemets, A. E., Batterberry, T. D., & Oliveira, E. P. (2010). U.S. Patent No. 7,725,557. Washington, DC: U.S. Patent and Trademark Office.

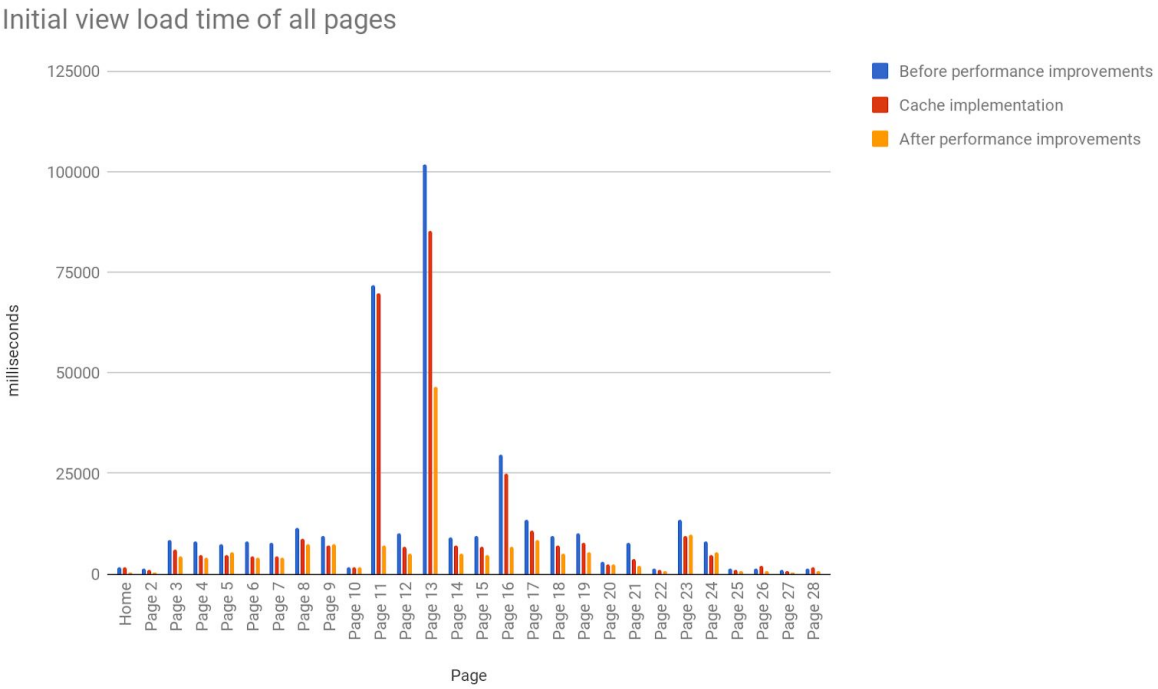
[16] Sleit, A. (2017, May). Evaluating IndexedDB performance on web browsers. In Information Technology (ICIT), 2017 8th International Conference on (pp. 488-494). IEEE.

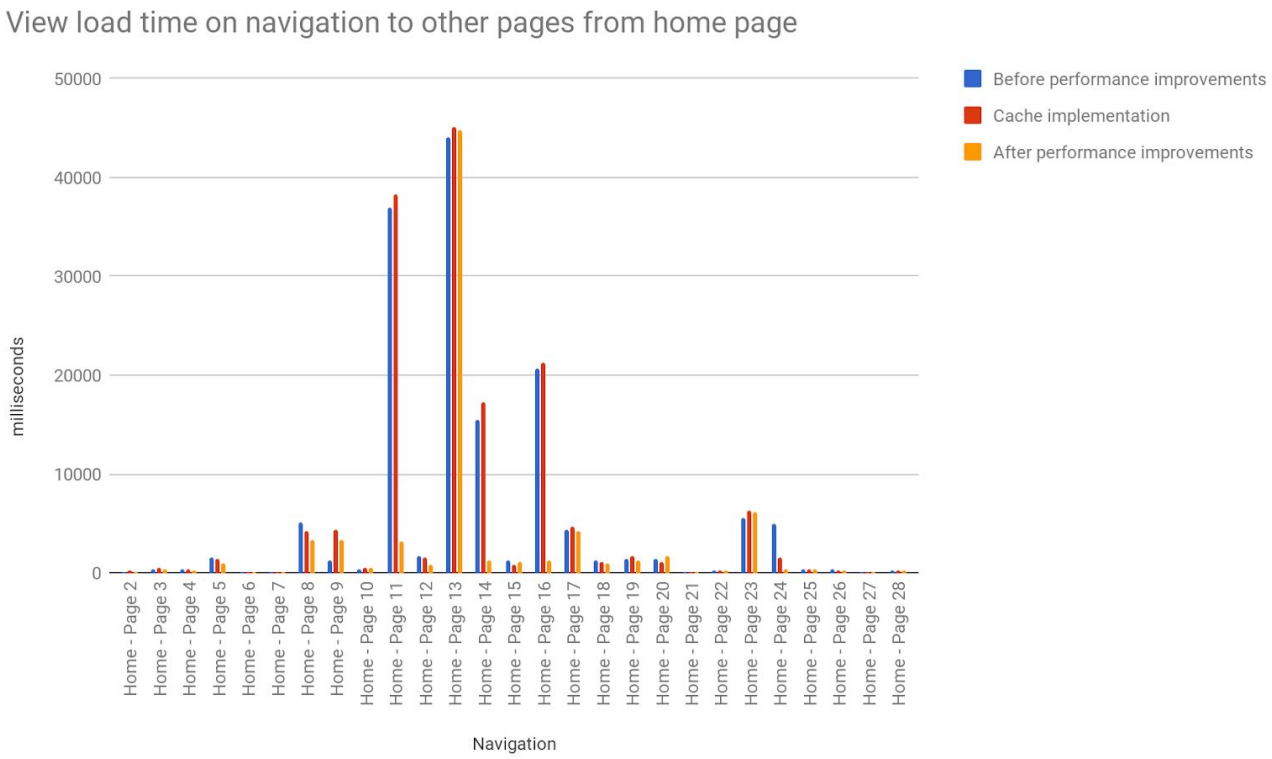
[17] Verdú, J., & Pajuelo, A. (2016). Performance scalability analysis of javascript applications with web workers. IEEE Computer Architecture Letters, 15(2), 105-108.

[18] Okamoto, S., & Kohana, M. (2011). Load distribution by using web workers for a real-time web application. International Journal of Web Information Systems, 7(4), 381-395.

APPENDICES

Appendix A: Median load time for each page





Median load time for each page

App load time of all pages	Home	Page 2	Page 3	Page 4	Page 5	Page 6	Page 7	Page 8	Page 9	Page 10	Page 11	Page 12	Page 13	Page 14	Page 15	Page 16	Page 17	Page 18	Page 19	Page 20	Page 21	Page 22	Page 23	Page 24	Page 25	Page 26	Page 27	Page 28
Before performance improvements	4071	4059	10903	10544	9939	10530	10234	14143	11848	3999	74037	12070	103960	11734	12060	32123	16001	11838	12524	5114	10025	3608	15586	10366	3828	3723	3770	3796
Cache implementation	4518	3728	8735	7624	7395	7345	7030	11572	10018	4419	72427	8920	88000	9428	9238	27353	13259	9537	10394	4858	6005	3465	11991	7188	3452	4461	3049	3872
After performance improvements	2473	2468	7065	6707	7793	6646	6711	9804	9964	4081	9434	7530	48730	7448	7141	9206	10781	7489	7696	4676	4362	2889	12187	7616	2779	2976	2507	2788
Initial view load time of all pages	Home	Page 2	Page 3	Page 4	Page 5	Page 6	Page 7	Page 8	Page 9	Page 10	Page 11	Page 12	Page 13	Page 14	Page 15	Page 16	Page 17	Page 18	Page 19	Page 20	Page 21	Page 22	Page 23	Page 24	Page 25	Page 26	Page 27	Page 28
Before performance improvements	1694	1518	8404	8077	7389	8040	7736	11486	9292	1668	71751	10011	101718	9198	9505	29767	13508	9391	10126	2901	7774	1311	13367	8121	1514	1513	1153	1499
Cache implementation	1869	1055	6198	4859	4595	4539	4316	8847	7202	1808	69788	6738	85191	6948	6780	24823	10744	7006	7754	2301	3595	1004	9514	4703	1117	1957	622	1543
After performance improvements	291	344	4462	4215	5326	4119	4154	7273	7266	1789	6935	5025	46462	5063	4624	6773	8405	5006	5382	2210	2161	655	9798	5554	593	824	308	657
View load time on navigation to other pages from home	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -	Home -
Before performance improvements	148	438	362	1510	61	24	5156	1223	447	36930	1780	44100	15527	1323	20696	4412	1258	1474	1401	14	201	5548	4963	410	386	85	266	
Cache implementation	173	527	325	1470	60	21	4245	4353	546	38330	1530	45160	17246	851	21209	4621	1185	1685	1046	13	201	6261	1536	306	196	42	247	
After performance improvements	130	357	292	1035	73	29	3373	3295	587	3156	818	44740	1200	1077	1332	4202	1043	1220	1708	10	193	6203	359	320	243	46	232	

Appendix B: User Experience Questionnaire (UEQ)

FPT UX Survey May 2018

With the help of the word-pairs please enter what you consider the most appropriate description for FPT. Please click on your choice in every line!

	1	2	3	4	5	6	7	
Annoying	◯	◯	◯	◯	◯	◯	◯	Enjoyable
Not understandable	◯	◯	◯	◯	◯	◯	◯	Understandable
Creative	◯	◯	◯	◯	◯	◯	◯	Dull
Easy to learn	◯	◯	◯	◯	◯	◯	◯	Difficult to learn
Valuable	◯	◯	◯	◯	◯	◯	◯	Inferior
Boring	◯	◯	◯	◯	◯	◯	◯	Exciting
Not interesting	◯	◯	◯	◯	◯	◯	◯	Interesting
Unpredictable	◯	◯	◯	◯	◯	◯	◯	Predictable
Fast	◯	◯	◯	◯	◯	◯	◯	Slow
Inventive	◯	◯	◯	◯	◯	◯	◯	Conventional
Obstructive	◯	◯	◯	◯	◯	◯	◯	Supportive
Good	◯	◯	◯	◯	◯	◯	◯	Bad
Complicated	◯	◯	◯	◯	◯	◯	◯	Easy
Unlikable	◯	◯	◯	◯	◯	◯	◯	Pleasing
Usual	◯	◯	◯	◯	◯	◯	◯	Leading edge
Unpleasant	◯	◯	◯	◯	◯	◯	◯	Pleasant

Secure	○	○	○	○	○	○	○	Not secure
Motivating	○	○	○	○	○	○	○	Demotivating
Meets expectations	○	○	○	○	○	○	○	Does not meet expectations
Inefficient	○	○	○	○	○	○	○	Efficient
Clear	○	○	○	○	○	○	○	Confusing
Impractical	○	○	○	○	○	○	○	Practical
Organized	○	○	○	○	○	○	○	Cluttered
Attractive	○	○	○	○	○	○	○	Unattractive
Friendly	○	○	○	○	○	○	○	Unfriendly
Conservative	○	○	○	○	○	○	○	Innovative

Appendix C: Standard deviations and margins of error of each navigation to other pages from home

| Before performance improvements | Home - P | Home - Home | Home - F | Home - I | Home - F | Home - I | Home - F | Home - I | Home - I | Home - Home | Home - P | Home - P | Home - P

Appendix D: Means of UEQ items from UX test 1 and 2

UEQ question number	Mean of each UEQ item (UX test 1)	Mean of each UEQ item (UX test 2)
0	0.9655172414	0.96
1	1.24137931	1.48
2	0.8275862069	1.08
3	1.206896552	1.44
4	2.103448276	1.708333333
5	0.4482758621	0.44
6	1.068965517	1.086956522
7	1	1.304347826
8	0.4482758621	0.4545454545
9	0.7777777778	0.04761904762
10	1.481481481	1.285714286
11	1.703703704	1.619047619
12	0.6785714286	1.095238095
13	1.148148148	1.142857143
14	0.6296296296	0.09523809524
15	1.153846154	1.2
16	0.6296296296	0.75
17	0.7777777778	0.85
18	1.321428571	1.368421053
19	1	1.157894737
20	0.9285714286	1.578947368
21	1.407407407	1.578947368
22	1.074074074	1.421052632
23	0.9259259259	1.105263158
24	1.37037037	1.526315789
25	0.5925925926	0.2105263158