Indonesian Journal of Computer Science
ISSN 2549-7286 (*online*)
Jln. Khatib Sulaiman Dalam No. 1, Padang, Indonesia
Website: ijcs.stmikindonesia.ac.id | E-mail: ijcs@stmikindonesia.ac.id

Website Optimization and Analysis on XYZ Website using Web Core Vital Rules

Kristian Handoko Wijaya Sukardjoh[1], Amalia Zahra[1]
kristian.sukardjoh@binus.ac.id, amalia.zahra@binus.edu
[1]Computer Science Department, Binus Graduate Program - Master of Computer Science, Bina Nusantara University, Jakarta, 11380, Indonesia

Article Information

Submitted : 24 Aug 2023
Reviewed: 17 Oct 2023
Accepted : 30 Oct 2023

Keywords

website, optimization, vital core web rules

Abstract

XYZ website is a business website that operates in the field of e-commerce which is implemented through websites and applications, for several years the website has had a percentage of users' usage speed which has decreased quite a bit and has become old, due to lack of maintenance of some of the features contained in the website application which have an impact on the lack of customer interest in buying goods on the XYZ website and more influential in terms of access from searching e-commerce notifications from Google that if the percentage of websites decreases over a long period of time, this will result in websites not being allowed to publish advertisements. In this study, we analyze the problem to understand the problem starting from small things, namely from the use of programming languages, the data provided, the use of writing code, third party or vendor support, filling out website content, and websites using vital core web architecture. So that the website used has good comfort, accelerates the use of the website which can be affected by the large number of customer visitors, and can facilitate the development team's performance in management and maintenance and provide many positive things from customers so that business runs fast and provides convenience for customers and the XYZ website can received by Google.

A. Introduction

In this study related to implemented through websites and applications. The process of designing a website using the programming language used is JavaScript, HTML, CSS, Library JavaScript React.Js, and Node.js combined as a page server so that SEO needs can be processed through server-side rendering. In the previous few years until now the performance of the website published the percentage of the speed of use of the user is quite declining and becomes long, due to the lack of maintenance of some of the features contained in the website application, it has an impact on the lack of customer interest to buy goods on the XYZ website and more than affect also in terms of access from the search engine from Google. The use of data for the website provided by the external team is quite a lot that is not related to customer needs. Not enough analysis of website performance because this website prioritizes business first so that this website business continues to run. For website optimization that is applied in terms of business.

This XYZ website also affects the assessment of Core Web Vital standardization. Core Web Vital is a support feature so that this website can be published by Google from search engines and advertising. This impact also affects the use of architecture models on the XYZ website, website modules that use a monolith flow system, which impacts the process of working on website systems designed in 1 bundle. In this study, to understand the problem from the start of small things, namely from the use of programming languages, the data provided, the use of writing code, third party/vendor support, filling the content with the website, and the website using the vital core web. This study uses code refactoring techniques that minimize code and make it easier for the team to analyze applications and perform rules from the vital core web, namely rules from Google to adjust Google's needs so that the website can have a better presentation and can be allowed to publish in advertising.

The problem formulation of this research can be explained as follows, what causes the XYZ website to have a fairly low speed, how to implement a website to make it faster, and how can the XYZ website will be allowed to work together by Google so that it can be published? The purpose of this research is to analyze the website to find out what factors cause the website to be lower in speed, examine the design of website optimization with appropriate standards so that the website speed is better, and provide a presentation of website optimization scoring to make it better and adjusts to the rules of Google needs. And also the benefits provided are to find out the causes of the problems faced by the XYZ website related to website speed, make the website faster and more structured which affects website optimization, and get cooperation from Google to publish the XYZ website so that business runs better.

The scope applied in the research is as follows More to the visual side and website performance. the scope includes the front-end team. The sample pages that will be tested are several website pages, namely Profile Page, Microsite Landing Page, Product Catalogue Page, Homepage, and Checkout Page. The analysis applied is influential in the architecture of the path in the XYZ Website.

B. Research Methods

1. **System Design** percentage of comparison tests and getting the total results of the comparison of Website Performance analysis on Website XYZ on react js using Google pageSpeed core web vital for illustration describes the sequence of work processes in this study. Website Performance is a type of testing to ensure the software will work properly under the expected workload. In focusing the main goal is to implement Performance Elimination. The focus of Performance Testing, namely: Speed, Scalability and Stability [13]. Performance testing is essential for analyzing and monitoring the performance of web applications. How page speed performance is affected, some of the most common factors that slow down page speed on a website page on the client side include scripts, fonts, and plugins (HTML, JavaScript, CSS). However, when a page request is made, the client-side and server-side components must complete their respective operations [17].

2. Research Design

2.1. Application Architectural Changes

Architectural changes from monolithic to microservice aim to have a more straightforward design and easier to monitor if there is a system failure in one of the system components.

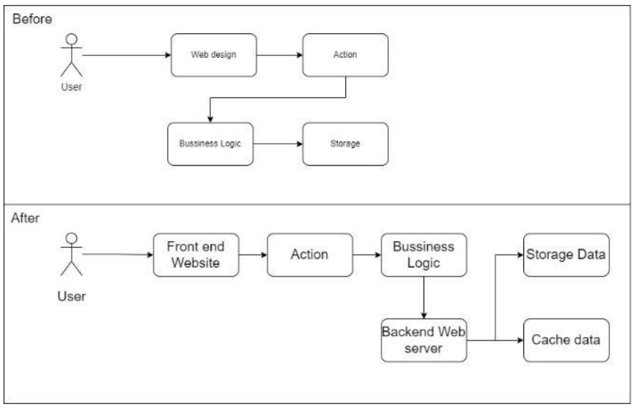


Figure 1. Architecture Microservices

Another benefit is that scalability can be applied to certain system components, to improve performance, the system needs to be maintained with sufficient resources.

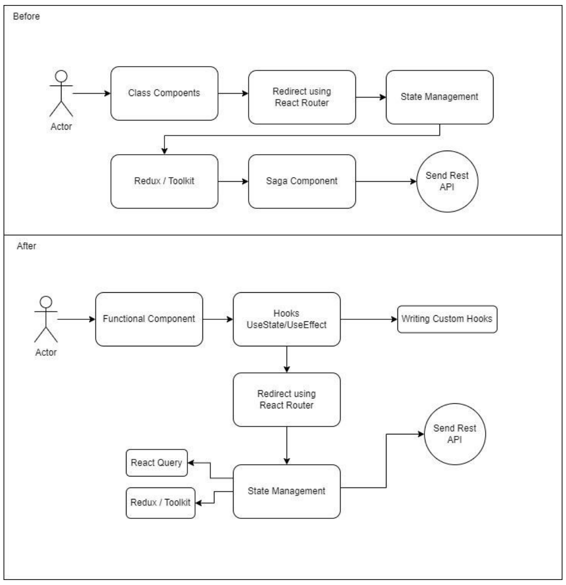


Figure 3. System Flow Functional

2.2. Change Code Package

To solve the Size of code problem, it is necessary to implement changes in writing the code package itself. The implementation of the solution includes rendering using Webpack to combine all scripts, this will compile our entire application into one file and then have an index file. The very basic HTML is limited to bundled JavaScript files. In writing code, apply solutions by cleaning up function/stateless components and React.pureComponent, using Webpack production markers, and applying immutable data structures, using React.Fragments.

2.3. Response Time & Page Load Time Management

Changes in architecture and code packages, it is expected to speed up the Response Time & page Load Time of the website. However, these changes must be continuously monitored to maintain Performance at an optimal state with the target {Response Time target} using the Google Page Speed tool with core web vital rules.

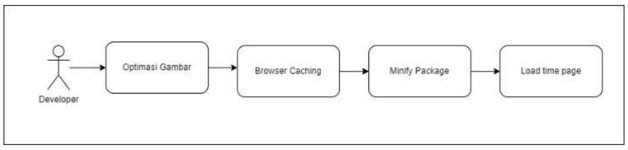


Figure 4. Flowchart Functional Response Time

2.4. Size of Code

The size of the code referred to in this study is the file size resulting from the rendering or compile process when you want to deploy to the server. To do rendering will use Webpack to combine all scripts. This will compile our entire application into one file. Once completed, then we can have a very basic index.html file that only creates bundled JavaScript files. After successfully rendering then the application can be deployed to the server.

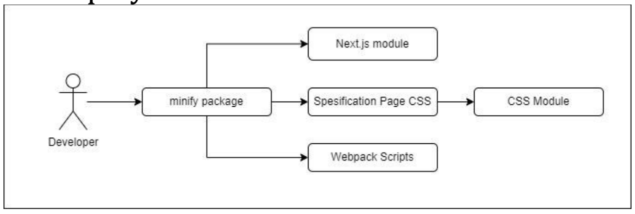


Figure 5. System applications

2.5. Response Time

Load Time and Response Time are the main factors for ranking pages for the purpose of faster access. Faster website access is a factor that increases the usability of web pages. response time is divided into several components, namely as follows.

1. Blocking The state when the browser is not ready to send a request.
2. DNS Lookup The state when the browser is looking for DNS information.
3. Connecting The browser is connecting to the server.
4. Sending The browser is sending data to the server.
5. Waiting The browser is waiting for data from the server.
6. Receiving The process when the browser receives data from the server. in seeing the factors of Website delays in covering the various components above, these components are applied so that it runs well. [18]

3. Implementation Code

Implementation code refers to converting an idea or concept design into real code in a particular programming language. It involves translating the planned solution or algorithm into instructions that can be understood by the computer. In the context of software development, code implementation is an important step in the software development cycle.

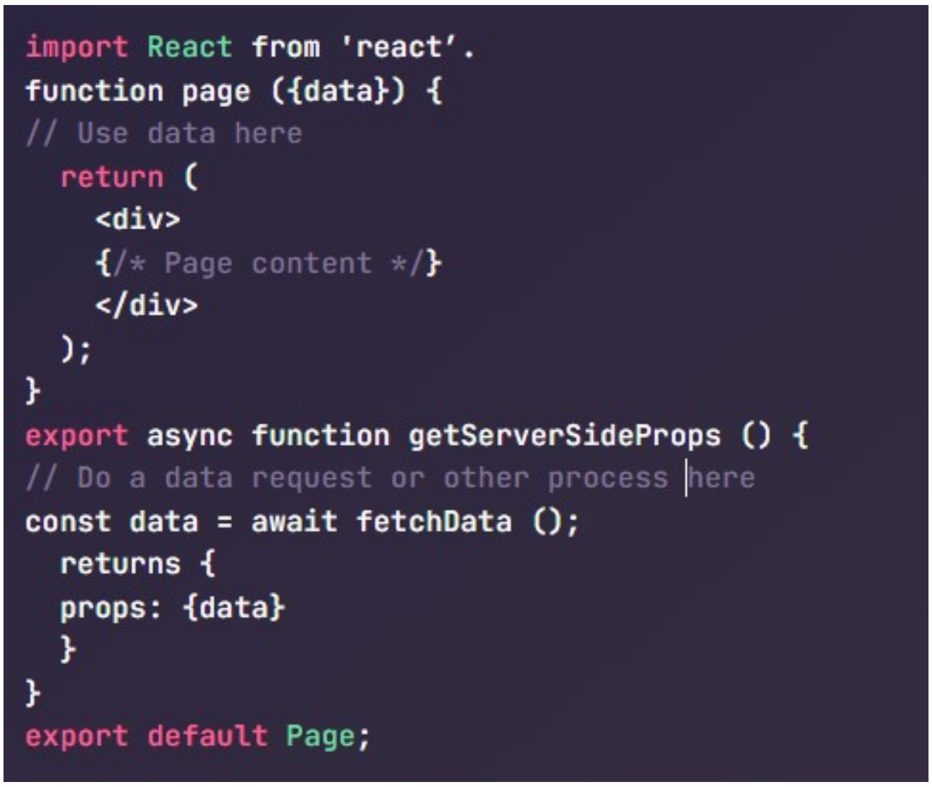
The code implementation process involves writing lines of code that conform to the design specifications and functional requirements of the software. At this stage, the developer selects a programming language suitable for the project, sets up data structures, implements algorithms, connects components, and takes care of various other technical aspects. A stage for how to implement the code to be designed:

3.1. Server-side rendering (SSR) implementation

Server-Side Rendering (SSR) is a technique in web development in which web page content is generated on the server side and then delivered to the client (browser). Here are the general steps for performing SSR in React using Next.js:

3.1.1. Make sure to have Node.js installed and run the following command in terminal.
"npx create-next-app project-name".

3.1.2. Enter the use of "getServerSideProps" which will be run on the server side before the page is rendered.



```
import React from 'react'.
function page ({data}) {
// Use data here
  return (
    <div>
    {/* Page content */}
    </div>
  );
}
export async function getServerSideProps () {
// Do a data request or other process here
const data = await fetchData ();
  returns {
  props: {data}
  }
}
export default Page;
```

Figure 7. Source Code Server-Side Rendering

3.1.3. Make sure to use API requests in server-side rendering according to the ssr section analysis.

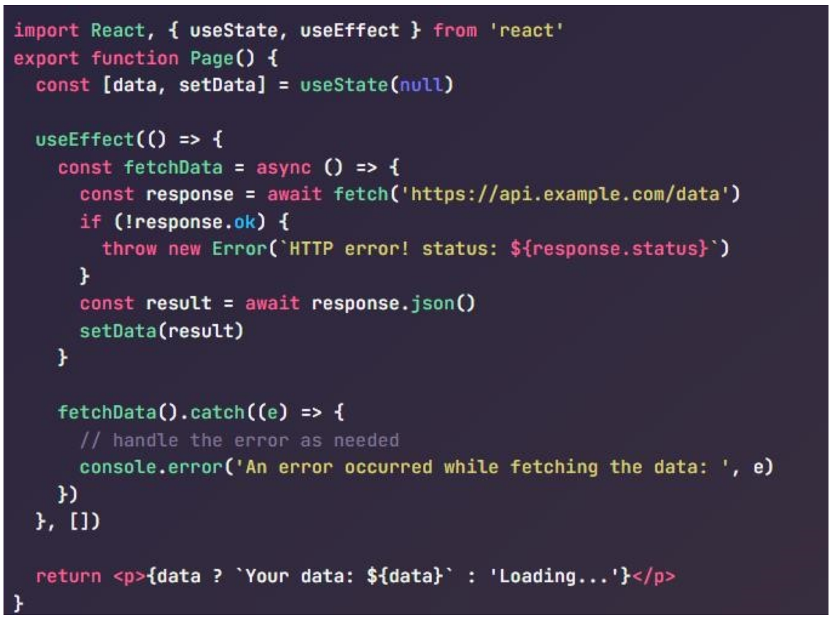
3.2. Client-Side Rendering (CSR) Implementation

Client-Side Rendering (CSR) is an approach in web development in which web page content is generated on the client side (browser) using JavaScript. This contrasts with Server-Side Rendering (SSR) where content is generated on the server side before being sent to the client. Here are the general steps for performing Client-Side Rendering (CSR) in React:

3.2.1. API and Data usage using methods like fetch or libraries like Axios. This data can then be processed and displayed in components.

3.2.2. Managing State Changes: When state changes, react will automatically update the appearance of the affected component. You can use the setState method to change state and trigger updates.

3.2.3. perform performance optimizations by avoiding excessive rendering and leveraging features such as React.memo to avoid unnecessary rendering. implementing the code as follows:



```
import React, { useState, useEffect } from 'react'
export function Page() {
  const [data, setData] = useState(null)

  useEffect(() => {
    const fetchData = async () => {
      const response = await fetch('https://api.example.com/data')
      if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`)
      }
      const result = await response.json()
      setData(result)
    }

    fetchData().catch((e) => {
      // handle the error as needed
      console.error('An error occurred while fetching the data: ', e)
    })
  }, [])

  return <p>{data ? `Your data: ${data}` : 'Loading...'}</p>
}
```

Figure 8. Source Code Client-Side Rendering

3.3. Image optimization on the website

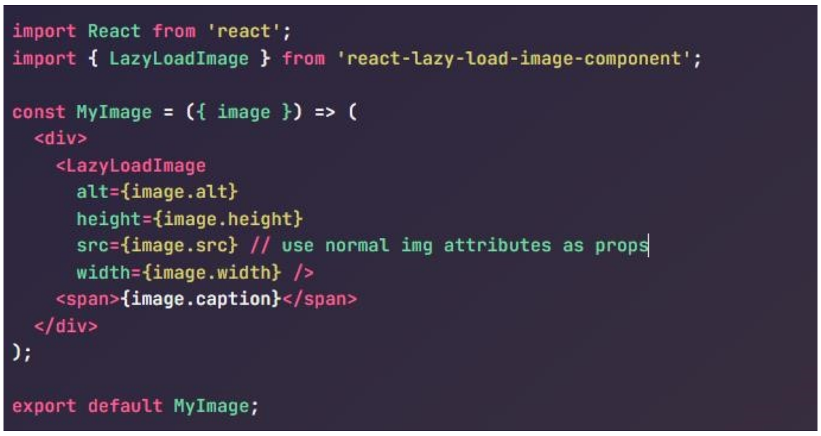
Image optimization is the process of reducing the file size of images without sacrificing quality. This can be done by using lossless or lossy compression, resizing images to the appropriate size, and choosing the right file format.

3.3.1. Use a CDN to deliver images to users from the server closest to their location.

3.3.2. Use the react-lazy-load-image-component library to optimize images and implement “react-lazy-load-image-component”.

3.3.3. Run the command in the terminal to install react-lazy-loadimage-component in the application to be designed. “npm i --save react-lazy-load-image-component”

3.3.4. The use of code that will be applied for image optimization in various sections. implementing the code as follows:



```
import React from 'react';
import { LazyLoadImage } from 'react-lazy-load-image-component';

const MyImage = ({ image }) => (
  <div>
    <LazyLoadImage
      alt={image.alt}
      height={image.height}
      src={image.src} // use normal img attributes as props
      width={image.width} />
    <span>{image.caption}</span>
  </div>
);

export default MyImage;
```

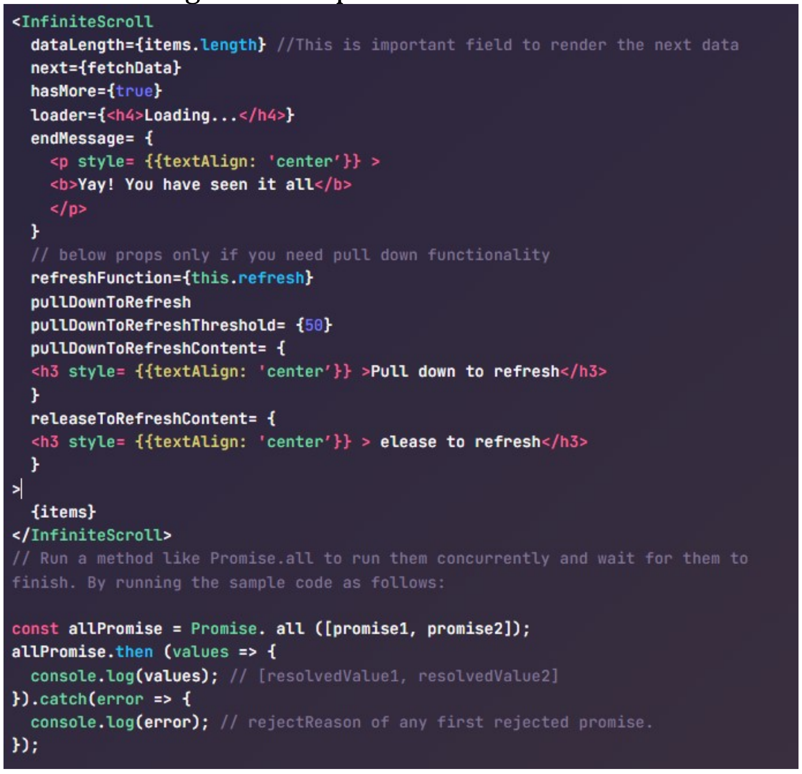
Figure 9. Source Code Imge Optimazation

3.4. Optimization of API data requests

Optimizing API requests in React.js projects is to improve application performance, reduce loading times, and minimize resource usage

3.4.1. Using the "react-infinite-scroll-component" library to get data restrictions in the amount of data retrieved at one time. This prevents overloading and reduces initial loading times.

3.4.2. Run the command in the terminal to install react-infinite-scrollcomponent in the application to be designed. "npm install --save react-infinite-scroll-component" Run the following code sample:



```
<InfiniteScroll
  dataLength={items.length} //This is important field to render the next data
  next={fetchData}
  hasMore={true}
  loader={<h4>Loading...</h4>}
  endMessage= {
    <p style= {{textAlign: 'center'}} >
    <b>Yay! You have seen it all</b>
    </p>
  }
  // below props only if you need pull down functionality
  refreshFunction={this.refresh}
  pullDownToRefresh
  pullDownToRefreshThreshold= {50}
  pullDownToRefreshContent= {
  <h3 style= {{textAlign: 'center'}} >Pull down to refresh</h3>
  }
  releaseToRefreshContent= {
  <h3 style= {{textAlign: 'center'}} > elease to refresh</h3>
  }
>
  {items}
</InfiniteScroll>
// Run a method like Promise.all to run them concurrently and wait for them to
finish. By running the sample code as follows:

const allPromise = Promise. all ([promise1, promise2]);
allPromise.then (values => {
  console.log(values); // [resolvedValue1, resolvedValue2]
}).catch(error => {
  console.log(error); // rejectReason of any first rejected promise.
});
```

Figure 10. Source Code React pagination

3.4.3. After getting the data that will be displayed in pagination, you can calculate the total number of pages and set the pageCount in the state according to that data.

3.4.4. Use currentPage to display data corresponding to the selected page.

4. Data Collection Method

4.1. Methodology

The collection method in this study includes some of the information needed by the results of the requirements by the Performance Website standardization using the following methods:

1. Field Research: In the field research method conduct research Observation.
2. Library research: In library research we conduct research based on books, articles and journals and sources related to the research we are doing

5. Analysis Technique

5.1. Related Research in Website Performance

Website performance has its own indicators so that the website can run smoothly and the performance of the website is good for users to use. First Contentful First or abbreviated as FCP is an assessment of the first time the browser runs the DOM element script, and First Input Delay or abbreviated as FID is a measurement report from users implementing client side or running actions from users [5]. Google PageSpeed Insight has 3 groups of website speed assessments, which are as follows - Fast (Green) with an assessment of more than 90.

- Average (Orange) orange color with an assessment of more than 50 and less than 90.
- Slow (Red) with an assessment of less than 50 (Xing, 2019).

Step by step for website performance:

1. JavaScript Response Code Standardization: To implement with the support of response code standardization in JavaScript [7].
2. Implementation of react.js support for Vital Core Web scoring : React.js is a JavaScript library that can help create responsive and interactive web applications. In order for the React.js application to support the performance [7].
3. Vital Core Web scoring on Google Pagespeed: Analysis of the implementation of the React.js: file structure is the flow of using the correct code from React.js so that the structure of using the file is tidier [7].

5.2. Standardization of revenue response data from APIs

1. Lightweight data formats like JSON or protobuf will speed up data load time and increase the speed of your application. Avoid using complex or heavy data formats like XML.
2. Data compression can reduce the size of transferred data and speed up data load time. You can use compression techniques like gzip or deflate.
3. Limit the amount of data sent by the API by sending only the data required by the application. Avoid sending irrelevant data or too much data in a single request[20].

5.3. Website performance conceptual measure development

Website performance measures the overall attractiveness from the consumer's point of view. The performance literature suggests that indicators cause performance and not the other way around. Formative rather reflective measures best assess website performance to define a construct, and suggest clarifying the assessing entities, objects, and attributes. In this case, travelers seeking information online are the assessing entities, websites that provide tourist information are the objects, and website performance evaluation is the attribute. To better understand the performance of the Website and further development of measurements, a qualitative survey was conducted. Data were collected through in-depth interviews.

5.4. Performance in Google pageSpeed insight

PageSpeed Insights is an application for analyzing website performance from Google. The problem with SPA (Single-Page Applications) is that they display content after loading a chunk of JavaScript first, so it takes a little time for the client before it can actually render the content and it can destroy PageSpeed Insights scores [23].

A commonly used method uses the PACE framework which has the following points:

1. Content Optimization from google optimization scoring
2. Page Speed Optimization
3. SEO Optimization (Search Engine Optimization)
4. User Data Analysis (Data response speed)
5. Use of microservices architecture model

5.5. Section Page Method

The analysis method that will be designed in this study using Linear Regression. This trial used a percentage of the standardization of websites in Indonesia with the XYZ Website. With data size from Core Web Vitals, namely Largest Contentful Paint (LCP), First Input Delay (FID), Cumulative

Layout Shift (CLS), First Contentful Paint (FCP), Interaction to Next Paint (INP), Time to First Byte (TTFB).

The sections on the website are divided into pages A to page H, each page has different content. In the analysis process, it will be divided into 2 types of flow chart components, namely general components, and individual components. 1. General components

General components are certain parts that are widely applied by all different pages and sections on the XYZ website. The use of general components helps make it easier to manage websites and achieve uniformity in appearance and behavior.

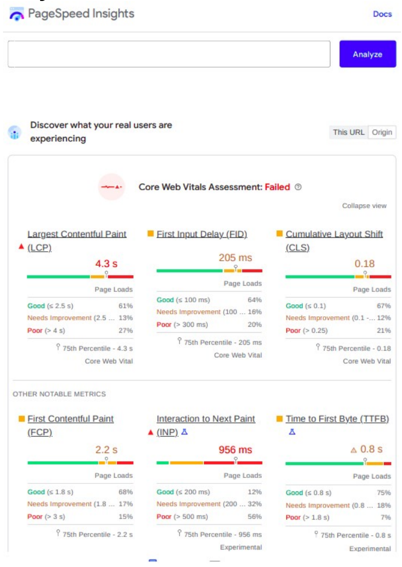
2. Individual components

Individual components are individual sections that apply to specific pages. The use of individual components helps make it easier to manage the website and achieve uniformity in appearance and behavior. This component is designed with certain functionality so that it is not dependent on other components.

C. Evaluation

Page Load Time is the time it takes to download and display all the content of a web page in a browser. The performance of this test will be carried out with the help of the Google Chrome performance tool to calculate the time, which starts when the button is clicked and ends when the rendering process is carried out. The tool that will be used to assess or test is Google PageSpeed Insight.

Using Google PageSpeed Insight is to access Google PageSpeed Insight on the browser, then enter the address of the Website you want to analyze [23]. To continue, please click the Analyze button as shown below:



Discover what your real users are experiencing

Core Web Vitals Assessment: Failed

Largest Contentful Paint (LCP)
4.3 s
Page Loads
Good (≤ 2.5 s) 61%
Needs Improvement (2.5 ... 13%
Poor (> 4 s) 27%
75th Percentile - 4.3 s
Core Web Vital

First Input Delay (FID)
205 ms
Page Loads
Good (≤ 100 ms) 64%
Needs Improvement (100 ... 16%
Poor (> 300 ms) 20%
75th Percentile - 205 ms
Core Web Vital

Cumulative Layout Shift (CLS)
0.18
Page Loads
Good (≤ 0.1) 67%
Needs Improvement (0.1 -... 12%
Poor (> 0.25) 21%
75th Percentile - 0.18
Core Web Vital

OTHER NOTABLE METRICS

First Contentful Paint (FCP)
2.2 s
Page Loads
Good (≤ 1.8 s) 68%
Needs Improvement (1.8 ... 17%
Poor (> 3 s) 15%
75th Percentile - 2.2 s

Interaction to Next Paint (INP)
956 ms
Page Loads
Good (≤ 200 ms) 12%
Needs Improvement (200 ... 32%
Poor (> 500 ms) 56%
75th Percentile - 956 ms
Experimental

Time to First Byte (TTFB)
0.8 s
Page Loads
Good (≤ 0.8 s) 75%
Needs Improvement (0.8 ... 18%
Poor (> 1.8 s) 7%
75th Percentile - 0.8 s
Experimental

Figure 11 Google Pagespeed website

After entering and analyzing our website URL into the column provided by Google PageSpeed Insight, we will get a performance report, including the Core

Web Vitals Assessment and the number of requests obtained by the website. For how to apply the core web vital rules themselves in using the load time system, you can see in terms of the list of core web vital.

This step continues with the contents of the results of the implementation that have been implemented based on the analysis. In the results that have been applied, do an analysis using the percentage of vital website measurements using the Google Pagespeed Insight tool, which consists of Largest Contentful Paint (LCP), and First Input Delay (FID). , Cumulative Layout Shift (CLS), First Contentful Paint (FCP), Interaction to Next Paint (INP), and Time to First Byte (TTFB).

The assessment is analyzed using pages based on section page A to page H with an average of frequently visited pages. This analysis is divided into various analyses, namely as follows an analysis of numbers in components based on percentages on each page the percentage analysis based on all pages is a page.

Calculation Section	: page to be analyzed
Percentage	: total data transmission using rest api and access weight in each component
Component	: name based on component.

The table below is a section presentation that has total pages based on the percentage of fetches using all page data:

Table 1. Percentage Section

Section	*Page A*	*Page B*	*Page C*	*Page D*
Percentage	20.9	24.4	11.2	16.8

Section	*Page E*	*Page F*	*Page G*	*Page H*
Percentage	6.7	2.9	8	9.1

Comparison of pages that have been implemented an analysis in calculating the Core Web Vitals Assessment based on the maximum and minimum in various rules:

Column explanation in each assessment:

Table 2. Column explanation

Column	Description
Average (AVG)	The average of each Core Web Vitals Assessment rule
Minimum (Min)	Minimum score for each Core Web Vitals Assessment rule
Maximum (Max)	Maximum value for each Core Web Vitals Assessment rule

Minimal page (Min P)	The selected page with the lowest Core Web Vitals Assessment rule
Maximum page (Max P)	The selected page with the highest Core Web Vitals Assessment rule

Before implementing code optimization website:

Table 3. Score Core Webvital Assessment before implementing code

CWV	***LCP***	***FID***	***CLS***	***FCP***	***INP***	***TTFB***
AVG	3.06	250.79	0.13	1.94	605.88	2.28
Min	*2.01*	*166.95*	*0.05*	*1.53*	*386.91*	*0.93*
Max	*3.94*	*348.59*	*0.19*	*2.47*	*781.20*	*2.91*
Min P	*Page G*	*Page C*	*Page F*	*Page E*	*Page A*	*Page B*
Max P	*Page H*	*Page B*	*Page G*	*Page B*	*Page F*	*Page F*

After implementing code optimization website:

Table 4. Score Core Webvital Assessment after implementing code

CWV	***LCP***	***FID***	***CLS***	***FCP***	***INP***	***TTFB***
AVG	1.67	77.62	0.14	2.47	89.10	0.44
Min	*0.26*	*53.93*	*0.01*	*2.08*	*1.99*	*0.01*
Max	*2.37*	*97.07*	*0.25*	*2.99*	*186.13*	*0.77*
Min P	*Page G*	*Page G*	*Page G*	*Page B*	*Page C*	*Page G*
Max P	*Page E*	*Page B*	*Page B* *Page C*	*Page F*	*Page G*	*Page F*

Comparison of Google rules an analysis of the scoring of all types of pages based on the Core Web Vitals Assessment with each page tested by the percentage of frequently visited pages.

Table 3.6 Description of core webvital assessments

Column	*LCP*	*FID*	*CLS*	*FCP*	*INP*	*TTFB*

Desciption	*Largest Contentful Paint*	*First Input Delay*	*Cumulative Layout Shift*	*First Contentful Paint*	*Interaction to Next Paint*	*Time to First Byte*

Before implementing code optimization website:

Table 8. Score Core Webvital Assessment before implementing code

Page	***LCP***	***FID***	***CLS***	***FCP***	***INP***	***TTFB***
page A	2.88s	243.35ms	0.15	1.96s	386.91ms	2.49s
page B	3.52s	348.59ms	0.12	2.47s	535.76ms	0.93s
page C	2.14s	166.95ms	0.08	1.59s	715.77ms	1.95s
page D	3.79s	340.86ms	0.18	1.61s	460.11ms	2.76s
page E	3.39s	213.52ms	0.13	1.53s	718.64ms	2.89s
page F	2.81s	187.62ms	0.05	2.38s	781.20ms	2.91s
page G	2.01s	201.45ms	0.19	2.09s	741.38ms	2.18s
page H	3.94s	303.96ms	0.14	1.89s	507.26ms	2.13s

After implementing code optimization website:

Table 9. Score Core Webvital Assessment after implementing code

Page	***LCP***	***FID***	***CLS***	***FCP***	***INP***	***TTFB***
page A	1.74s	88.29ms	0.11	2.32s	128.63ms	0.65s
page B	2.34s	97.07ms	0.25	2.08s	172.67ms	0.64s
page C	2.14s	72.39ms	0.25	2.69s	1.99ms	0.65s
page D	2.32s	73.87ms	0.17	2.54s	9.79ms	0.62s
page E	2.37s	92.64ms	0.07	2.66s	73.33ms	0.09s
page F	1.29s	54.59ms	0.04	2.99s	116.48ms	0.77s
page G	0.26s	53.93ms	0.01	2.35s	186.13ms	0.01s
page H	0.87	88.14ms	0.18	2.09s	23.77ms	0.10s

Assessment as a status percentage of scores from the Core Web Vitals Assessment A table that contains a result status that has been assessed by the Core Web Vitals Assessment.

Table 10. Description of status core webvital assesment

Status	*Desciption*
Good (G)	The results of the design were successfully implemented and can be accepted by google, the website is faster and lighter
Need Improvement (NI)	The design results are almost successful and not fails, can be accepted by Google and the website is stable.
Poor (P)	The results of the design fail, less accept by google and unstable website

Before implementing code optimization website:

Table 11. Score Core Webvital Assessment before implementing code

Page	***LCP***	***FID***	***CLS***	***FCP***	***INP***	***TTFB***
Page A	NI	NI	NI	NI	NI	P
Page B	NI	P	NI	NI	P	NI
Page C	G	NI	G	G	P	P
Page D	NI	P	NI	G	NI	P
Page E	NI	NI	NI	G	P	P
Page F	NI	NI	G	NI	P	P
Page G	G	NI	NI	NI	P	P
Page H	NI	P	NI	NI	P	P

After implementing code optimization website:

Table 12. Score Core Webvital Assessment after implementing code

Page	***LCP***	***FID***	***CLS***	***FCP***	***INP***	***TTFB***
Page A	G	G	NI	NI	G	G

page B	G	G	NI	NI	G	G
page C	G	G	NI	NI	G	G
page D	G	G	NI	NI	G	G
page E	G	G	G	NI	G	G
page F	G	G	G	NI	G	G
page G	G	G	G	NI	G	G
page H	G	G	G	NI	G	G

D. Conclusion

Website performance is very important because it has a direct impact on user experience, conversion rates, and brand image. Website speed can have an impact on conversion rates, such as sales or other desired actions. Users who must wait too long for a page to load are more likely to leave the site before taking the desired action. and A positive experience with a website can increase customer satisfaction, which in turn can lead to long-term support and loyalty. Thank you to your supervisor for supporting this journal and thank you to your supporters who were able to help complete this journal.

E. References

[1] Rompis, A. C. (2018). Perbandingan Performa Kinerja Node.js, PHP, dan Python dalam Aplikasi REST. CogITo Smart Journal, 171.

[2] Dimas, F. (2022). Analisis faktor - faktor yang mempengaruhi penggunaan website di PT.XYZ

[3] Ilham, Y. (2021). Analisa perbandingan performa penggunaan reactjs dan angularjs pada front-end website

[4] Dimas C. Y. N. (2021). Optimasi keamanan pada website menggunakan web application firewall (waf) dan hardening broken authentication.

[5] Fredy F. (2020). Pengaruh website design, website security, information quality, dan perceived ease of use terhadap customer satisfaction serta online purchase intention pada e-commerce indonesia di jakarta.

[6] SUDIRMAN, E. (2021, agustus 5). Content Delivery Network (CDN) Content Delivery Network.

[7] Voutilainen, J. (2017). Evaluation of Front-end JavaScript Frameworks for Master Data Management Application Development. Retrieved from theseus

[8] Wang, X. Y. (2009). The impacts of brand personality and congruity on purchase intention: Evidence from the Chinese mainland's automobile market. Journal of Global Marketing, 199-215.

[9] Application, W. E. (2018). React Framework (Creating a Web Application with React Native). International Journal of Recent Trends in Engineering and Research, 642–646.

[10] Arifin, Z. (2009). Evaluasi Pembelajaran. Remaja Rosda.

[11] Ariona, R. (2013). Tutorial fundamental dalam mempelajari HTML & CSS.

[12] Auler, R. B. (2014). Addressing JavaScript JIT engines Performance quirks: A crowdsourced adaptive compiler. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2018-237.

[13] Belanche, D. C. (2012). Website usability, consumer satisfaction, and the intention to use a website: the moderating effect of perceived risk. Journal of retailing and consumer services, 124-132.

[14] Chavare, N. N. (2016). Performance Comparison and Evaluation of jQuery with AngularJS. International Research Journal of Engineering and Technology.

[15] Constantinides, E. (2005). Influencing the online consumer's behavior: the Web. Internet research, 111-126.

[16] Destiningrum, M. &. (2017). Sistem Informasi Penjadwalan Dokter Berbassis Web Dengan Menggunakan Framework Codeigniter (StudiKasus: Rumah Sakit Yukum Medical Centre). Teknoinfo, 30.

[17] Dimas. (2021, Juni 28). Ini 10 Cara Meningkatkan Keamanan Website! Retrieved from exabytes: https://www.exabytes.co.id/blog/carameningkatkan-keamanan-website/

[18] Flavia´n, C. G. (2006). The role played by perceived usability, satisfaction, and consumer trust on Website loyalty. Information & Management, 1-14.

[19] Guha, A. S. (2010). The essence of JavaScript. Lecture Notes in Computer Science (Including Subseries Lecture Notes in 59 Artificial Intelligence and Lecture Notes in Bioinformatics), 126–150.

[20] KhuatKhuat, T. (2018, januari 1). Developing a frontend application using ReactJS and Redux. Retrieved from core: https://core.ac.uk/download/pdf/161432422.pdf

[21] Kumar, A. &. (2016). Comparative analysis of angularjs and reactjs. International Journal of Latest Trends in Engineering and Technology, 225227.

[22] Limbu, Y. B. (2018). The determinants and conSequences of Website credibility in e-retailing: examining the roles of ethical issues. International Journal of Electronic Marketing and Retailing, 89-108.

[23] Mubarok, I. (2019, maret 18). Cara Gunakan Google PageSpeed Insight untuk Optimasi Website. Retrieved from niagahoster: https://www.niagahoster.co.id/blog/google-pageSpeed-insight/

[24] Nayoan, A. (2021, agustus 23). Cara Mempercepat Loading Website dan Blog Anda. Retrieved from niagahoster: https://www.niagahoster.co.id/blog/caramempercepat-loading-blog/

[25] Nurmi. (2014). Membangun Website Sistem Informasi Dinas Pariwisata. Jurnal. Jurnal Edik Informatik, 1-6.
[26] Park, C. H. (31). Identifying key factors affecting consumer purchase behavior in an online shopping context. International journal of retail & distribution management, 2003.
[27] Prasetio, N. Z. (2021, April 7). Alasan Loading Website Harus Cepat Dibawah 3s dan Juga Aman? Retrieved from Exabytes: https://www.exabytes.co.id/blog/alasan-loading-website-haruscepat/#Alasan-Loading-Website-Harus-Cepat
[28] Putri, M. A. (2018). Performance Testing analysis on web application: Study case student admission web system. Proceedings - 2017 International Conference on Sustainable Information Engineering and Technology, 1-5.
[29] Ramos, M. V. (2018). AngularJS Performance: A survey. IEEE Software, 72-79.
[30] Riyanto, S. (2009). Membuat Web Portal Multi Bahasa Joomla 1.5X + CD. (1st Edition). Jakarta: Elex Media Komputindo.
[31] Rompis, A. C. (2018). Perbandingan Performa Kinerja Node.js, PHP, CogITo Smart Journal, 171.G. Smith, “Paper Title” (to be published).