Methods and Techniques to Improve Performance of Modern Web Applications using Next JS

Yudhajit Adhikary

Research Paper

Nov 2024

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

JS…….. JavaScript

DOM……Document Object Model

HTML……Hypertext Markup Language

CSS……Cascading Style Sheets

SSR……Server Side Rendering

SSG……Static Site Generator

CSR……Client Side Rendering

LCP……Largest Contentful Paint

FCP…….First Contentful Paint

CLS……Cumulative Layout Shift

FID……First Input Delay

UI…….. User Interface

INP……Interaction to Next Paint

TTFB……Time to first byte

SEO……Search Engine Optimisation

TCP……Transmission Control Protocol

HTTP……Hypertext Transfer Protocol

IP……Internet Protocol

CPU……Central Processing Unit

CSSOM……CSS Object Model

CDN……Content Delivery Network

TBT……Total Blocking Time

OS……Operating System

Abstract

Website performance is the most critical aspect when it comes on creating a business.It has a huge impact on the experience user will be having , how much your website can convert user , what user thinks about your brand , website page speed have a huge impact on sales , activities and other services. User who have to wait for long time for site to load are more likely to have bad impression and that leads to leaving the site before performing the desired action. Good website performance create a positive experience which increases customer satisfaction which finally leads to loyalty and long term support. Economic Times achieved an overall 43% better bounce rate across the entire website by optimising the website performance. The number of visitor has improved by an average of 22 percent for Renault domains which is from 51% to 73%. With every second delay on their site , BBC have experienced lost of 10% of it's users. By improves the page-speed of their website, Rakuten 24' increased conversion rate by 33.13% and revenue per visitor gets increased by 52.37%. Vodafone found that improvement on their website page speed leads to 8% more sales , 15% uplifting in the visit rate, 11% uplift in the cart to visit rate . RedBus improved their website page speed which resulted increase in sale by 7%. So knowing the real cost of website that failing to engage, retaining users or not able to meet customer expectations makes us more curious about how we can follow few fundamentals for improving performance and SEO of the website. Next JS is a react framework for building web application, it under the hood , comes with many configuration toolings like bundling , compiling and more, so instead of spending time with configuration , developer can spend more time on building the application. In this research paper we will discuss how we can leverage the builtin optimisation features provided by Next JS to improve the performance of the website .

CHAPTER-1

INTRODUCTION

1.1 Background

Optimising the user experience quality leads to long term success for any website , whether you are in business, marketing or development.User using browser are on a journey , and their each action is like the steps, so like real world, delays are nothing but distraction from their activity, and this may leads them to make mistakes . So if our website is slow and non responsive it may lead to lower satisfaction and user will escape from the site or whole journey. A page loads more often when the user are keen to learn about something new like any recent affairs , a new product etc. So from user's prospective since they have not yet achieved their goals they may be less tolerant to the website delay. The effect of website delay varies and it's hugely depends on user characteristics , past experience and the task urgency . According to Galletta, D. F., Henry, R., McCoy, S. & Polak, P. (2004) delay that can decrease satisfaction and intention to return on the any random site is 2 secs . According to Hoxmeier, J. A. & DiCesare, C. (2000) if we delay loading each panel navigation from 0 to 3 seconds satisfaction drops and when it drops from 9 to 12 seconds the intension to return also dropped. So we can say 6 seconds delay is enough to declare a website as slow. Oulasvirta, A., Tamminen, S., Roto, V. & Kuorelahti, J. (2005) says that mobile user didn't keep their attention on the screen for more than 4-8 seconds, so we can say a 5 second delay will seem 10 sec delay for a mobile user .

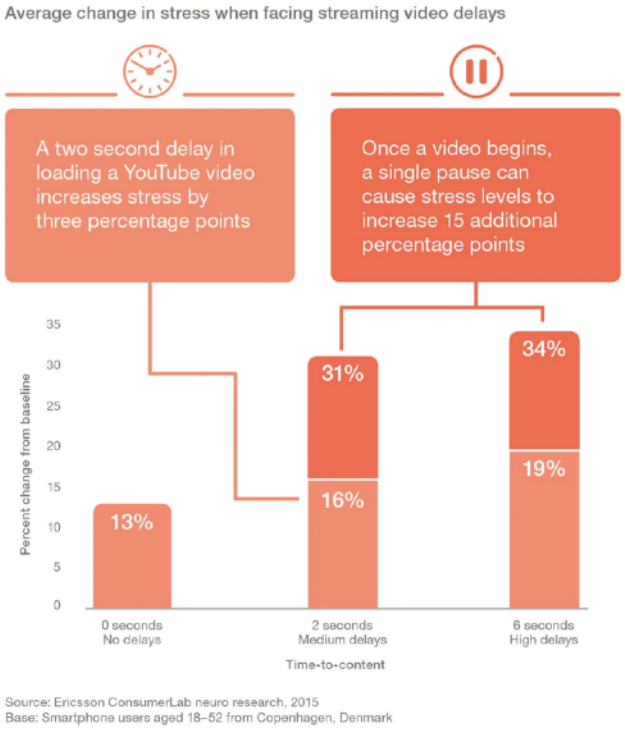


Figure 1: Increase of stress level of users due to delay of two second. (Ericsson 2016)

Card, S. K., Robertson, G. G., & Mackinlay, J. D. (1991) suggested that the speed of a system's response should be comparable to the delay human's experience when they interact with another, a response should take around 1- 4 seconds .

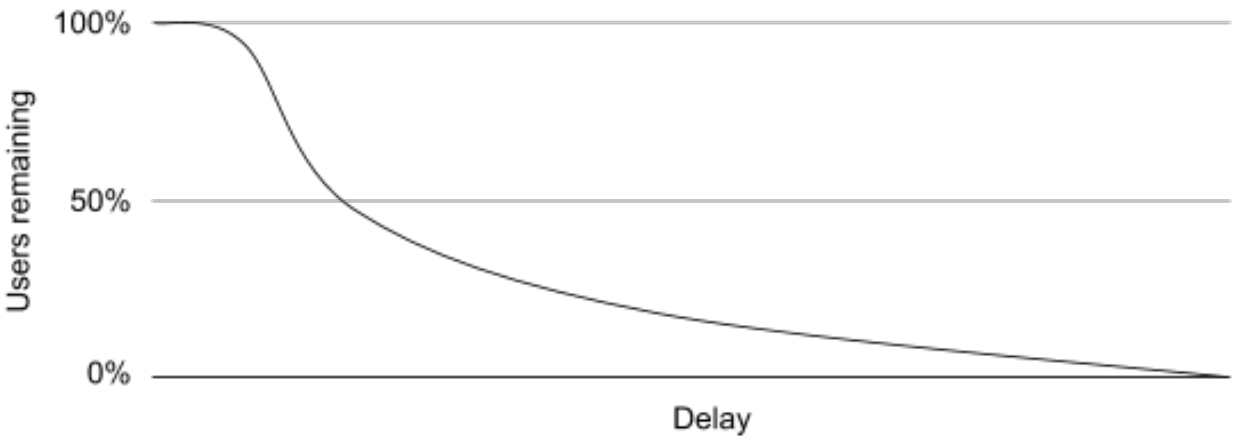


Figure 2: Graph of website delay vs users escaping website(Amar Sagoo , Annie Sullivan, Vivek Sekhar 2020)

An application speed depends on how long it takes to provide all the website related content and data to the client in first circle. Application execution collapses when server sends some render blocking resource during this circle . In this paper we will implement and analyse how we can use Next JS to improve the performance and search engine optimisation of a website by optimising those blocking resources.We will also discuss how we can implement those Next JS features practically in our code to improve the performance of a website. In this research paper, we will be analysing how Next JS can improve the page-speed of a website. For that first we will be analysing what are the factors that affects the page-speed of a website .Then we will explore and analyse how Next JS can leverage the page-speed of a website. After that we will be conducting a page-speed scan on a reference static site using Google page-speed insights tool , it is a tool used for improving the quality of web pages and is able to run a variety of test against a webpage while monitoring various performance like the core web vitals and speed index. Then after analysing the scan report we will be developing the same application using Next JS and will conduct another page-speed scan on that application . Finally we will compare and evaluate both the page speed scan report and will try to draw a conclusion on how Next JS is impacting the page-speed of the application .

1.2 Problem Statement

Web performance is a very critical aspect of web development, by optimising our website we are giving our user a better experience which help us in increasing traffics of our website and finally leads us to business improvement. Slow website will always have a negative impact on revenue , where as a fast site will always increase conversion rate and improves business outcomes . According to Ericsson(2016) the amount of stressed people gets while waiting for the delay in page speed is equivalent to that of watching a horror movies or solving a mathematical problem and greater than waiting in a checkout line at a retailer.

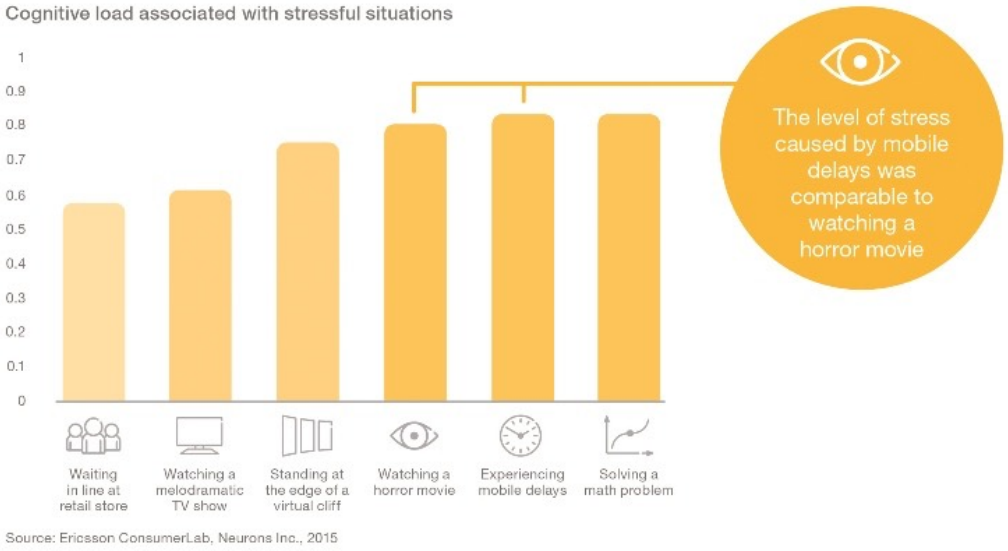


Figure 3: The level of stress caused by mobile delay (Ericsson , 2016)

When a user enter a url in the browser it sends GET request to the server to fetch it ,the HTML resource is requested first for a web page , so to ensure a good page speed of our website we should always ensure the HTML arrives quickly with minimal delay. Mainly there are two types of resource that blocks a HTML to arrive quickly, render blocking resource and parser blocking resource . In this paper we will discuss and analyse how Next JS , a react js framework efficiently handle those resources by applying it's builtin optimisation techniques , and how we can leverage those features technically to improve website performance.

1.3 Research Question/Aims and Objectives

In current scenario , maintaining a good page speed of a website across all browsers and device are very important to stay ahead in business competition , through website we used to showcase our products , our business ethics , brand value and loyalty to our customer , whether we are in business , marketing or development optimising a good page speed will always keep our website and our business ahead of others. The main aim of this research is to evaluate the effectiveness of Next js for improving performance of the website. The objectives or the steps to achieve the aim of the study are as follows:

a> To identifying factors which are responsible for slowing down the rendering process of the website .

b> To identify the optimising properties of Next JS which can help to optimise the performance of the website.

c> To draw conclusion based on the findings of the study.

1.4 Significance /Scope of the Study:

As modern web application grows , it becomes increasingly difficult to maintain their performance , that is why it is very essential to optimize application performance, by optimising our website we are giving our user a better experience which will help us with increasing traffic and finally leads to business improvement. Now let's discuss the scope of the research paper.

a> In this research paper we will be developing a real world application (https:// yudhajitadhikary.github.io/Portfolio) using React JS .

b> We will be generate performance report of React Application using google page-speed insights.

c> We will be measuring the page speed of the site using metrics defined by google core web vitals.

d> We will go through the report and will identify the loop holes present on the site which are affecting the page speed and will go through recommendation provided by google for improving the page-speed of the application.

e> We will developing the same application using Next JS and will try to apply those recommendation using Next JS.

f> Finally after applying the Next optimisation strategy on the application we will be generating the performance report of the optimised website for comparing and analysing the impact of Next JS on improving the page-speed of the website.

In this paper we will evaluating our website performance using google pagespeed insights, and will follow the core web vitals standards and recommendations provided by google.We will also consider the metrics defined in core web vitals as performance measuring parameters of the website . This paper will be written based on the pagespeed report generated in google page speed insights.

1.5 System Architecture Design:

The Layout of the next application can be divided into three sections:

1. Network Layer
2. Next Application instance: the SSR
3. External Service

1. **Network Layer:**

Network layer used to decide how the request will be handled by it's server and which protocol will be suitable to be utilised , When an application is requested by user, first the request goes to load balancer, load balancer check the request if the request is for loading website HTML content , it will send the request to the application server , else if the request is for loading static content like images, fonts , or a build file it will send the request to CDN and content will load from the CDN by this way we can reduce the static request load on the application server using load balancer to deliver all the static content from the edge location.

2. **Next Application instance: the SSR:**

This section determines how the application code should work , all request go through the next js custom cache server , which used to cache all incoming requests. When a request comes for the page mostly in server side rendering the next application loads all of the material and generates a HTML version of the page request. If we want dynamic data to be loaded with the HTML we can use Next Js implementation like getServerStaticProps, getInitialProps or getStaticProps. NextJs comes with Next Image which is a loader that transcript the image into a certain size or format to minimise the time of transmission over network , we can use webp format image and avif images as well to optimise the image size of our application.If we want to serve dynamic static content or if we need to store a sitemap that is very dynamic we need to store it alongside the website itself , we can use file request for dynamic routing .

3. External Service:

When user enter the website url on browser , it sends the first initial page request to the server , When load balancer receives the request it will route the request to custom server provided by Next application. It will check whether the file is already been cached or not , if it's cached , it will return the cache data, and if it is not cached ,it will load the next application and will run the server executable like GetStaticProps , GetServerSideProps and GetInitialProps to loads the complete next application and convert it to HTML . Then it caches the HTML file that has been parsed , and return the data to the client .Then the HTML markup is downloaded and starts loading in the browser , while loading it sends a request to render all the essentials like Javascript , Fonts , Images , CSS etc to load balancer. Load balancer will send the request to CDN and static storage bucket , we can also create a sub domain and connect the CDN to storage bucket directly bypassing the application load balancer. The website will initiate loading in the client side while all of them have been loaded , and if any frontend api needs dynamic loading , parallelly it can perform the calling and storing the data in state management system like redux or mobx. Now last step is loading the next page, when user moves from page A to page B , the Javascript required to load Page B will be preloaded and api request to get the content like getStaticProps, getInitialProps and getServerSideProps function will be initiated. If it is successful, the page B will start displaying with all the scripts and content to make it interactive.

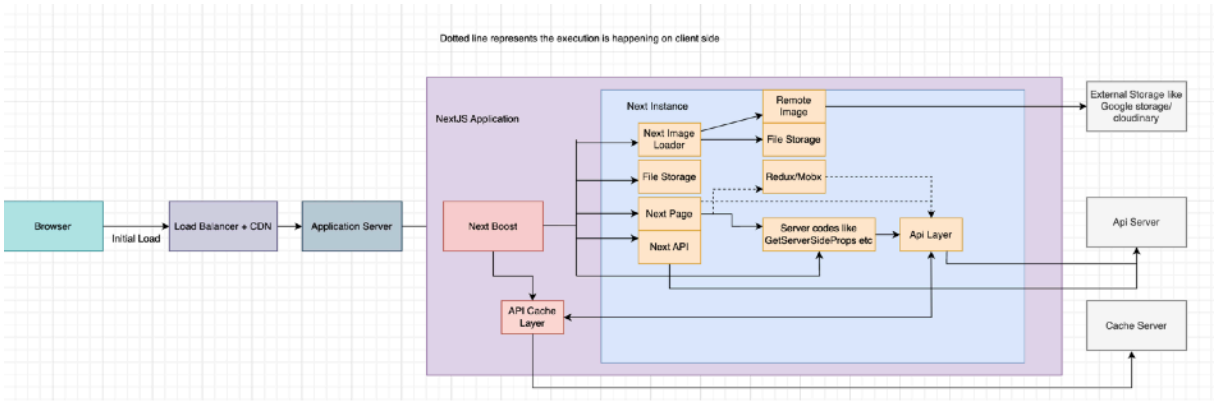


Figure 4: Next.js system architecture design

According to Sushin Pv(2022) the major reason for utilising Next JS is to have a default support for SEO , we can use Next JS script and can use different strategy to load the required third party scripts , while developing a page we should always keep our eyes on the build size , loading properly sized image and prefetching them impacts the page speed of the site a lot , we can use next image , it is a lazy loading technique to load the image , it is also vital to load images which are on the first fold of the website , like if we have 300*300 px in canvas , downloading 1200*1200 px will be waste of internet bandwidth.We can use dynamic import for lazy import which will prevent all unnecessary code from being bundled together. Next JS preloads all the JavaScript for pages having a link tag to them from current page , this speeds up the client side rendering , but if we are having a link in footer , user more likely will not click on those links , so we can mention prefetch=false on those link tags to prevent them from been preloaded . Next Js also provides builtin support for AMP, which increases page speed by creating high value page to AMP and make them load more faster.

CHAPTER-2

LITERATURE REVIEW

2.1: Importance of page performance

According to Zahra, Amalia, Handoko Wijaya Sukardjoh, Kristian, Khatib Sulaiman Dalam No, Jln (2023) website performance has a direct impact on brand image , conversion rate , and user experience. Website speed have a huge impact on sales and other desired actions. User are more likely to leave the site , if they have to wait too long before taking the desired actions. In order to increase customer satisfaction, loyalty and long term support , a positive experience on the website is very essential .

According to Galletta, D. F., Henry, R., McCoy, S. & Polak, P. (2004) the effects of increases in delay also appears to be dependent on the familiarity of the website.In familiar website user mostly use to complete their desired activity by tolerating the delay of the website. For unfamiliar website since user used to face navigation difficulty in unknown environment, so user are still tolerant to the delay they experienced .So they concluded that as user becomes more familiar with website , delay becomes more significant factor in formulating on intention and attitude.

According to Hoxmeier, J. A. & DiCesare, C. (2000) if the response time increases the user satisfaction decreases . User considers a system fast enough if it’s response time is within 6 second, their perceptions drops noticeably after nine second , which makes the application slow from user’s prospective.There are many factors which can result delay on website load on browser. Further research and usability studies may lead to a new understanding of response time guidelines for browser-based applications.

Kaaresoja, T., Brewster, S., & Lantz, V. (2014) suggests that tactile feedback latency should be between 5 and 50 ms, audio feedback latency between 20 and 70 ms, and visual feedback latency between 30 and 85 ms. Using these values will ensure that users will perceive the feedback as simultaneous with the finger's touch. These values also ensure that the users do not perceive reduced quality. In this research paper we will explore how we can utilise the Next JS optimisation techniques to reduce response and interaction time of the website . According to Ericsson (2016) delay in website or loading video when user in under time

pressure makes user heart rate increased by 38 percentage in average. 6 second delay to video increases the stress level by a third . The stress that user get for website delay is equivalent to giving maths test, watching a horror movies alone , and is greater than stress experienced by standing in the edge of virtual cliff.

Vasilije Vasilijević, Nenad Kojić, Natalija Vugdelija (2020) suggests the data present of the website should be easily accessible and searchable across all devices. We can optimize the SEO of our website by improving the quality of search which again depends on techniques and technology used and quality of code written .The main measuring parameter of website performance are speed, scalability , stability. Performance testing is something we should keep checking and monitoring continuously to ensure stable and consistent performance of our application. In this research paper we will explore how Next Js helps in increasing the organic traffic of the website by increasing the Seo score.

According to Azeez, Abdulsamod, Ayodeji, Bolaji (2022) majority of the website does not score up to 90% good score for each metrics define in lighthouse performance grading used and number of warning depicted for each metrics are mostly are rendering blocking resource , website not having any next generating formats, excessive DOM size, insufficient cache policies , unused css and JavaScript , large JavaScript file , network payloads , no usage of valid image alt attributes , colour contrast etc . In this research paper we will see how we can improve the performance of the website by resolve the above mentioned factor which affects website performance using Next JS.

2.2: Process of website rendering

According to Bangzhong Cao ,Minyong Shi , Chunfang Li (2017) when user enter the access url in the address bar of the browser , the browser checks for the IP address in the domain name resolution library. The client browser sends the request url to the server. When server receives request to generate HTML from the browser , the server sends response to the browser request. Browser accepts a parsed data send by the server , server also returns the related resource file to the browser. Finally browser will be parsing the related resource file and render the page from top to bottom. Now let’s try to figure out the waiting time involved in this whole process , those are the time when server are resolving domain for the server IP also know as domain resolution time , time required by a client to establish the

connection with the server , time taken by the TCP to establish a connection , time server takes to respond to the HTTP request and the time required to download the request content after the server respond to the request .The most common factor to slow down page speed on a web page includes scripts, fonts and plugins (html, Javascript and css).

According to Mariko Kosaka (2018) many activities or process used to happen under the hood on the process of rendering a website . Renderer process is responsible for everything that is happening inside a browser tab . Main thread mostly have to parse most of the code we send to the server. There are two other threads called compositor and raster thread which helps to make the render process efficient and smooth. The main job of renderer process is to convert JS ,CSS and HTML into a webpage that can interact with the user. In this research paper we will explore how Next JS helps to reduce the load of main threading by reducing the render blocking and parser blocking resources of the application.

2.3: Core Web Vitals

Amar Sagoo , Annie Sullivan, Vivek Sekhar (2020) have analysed millions of page report , impression and try to understand the impact of web vital metrics and threshold affect on the overall performance of the website, accordingly to them the website which meets the threshold defined in core web vital have 24 % less user who are most likely to abandon page loads, For news or shopping websites growth of their business is indicated by task completion percentage and traffic percentage. The news website which can achieve the threshold values of core web vitals are having 22% less abandonment of their website , and for shopping site , the percentage is 24% less abandonment rate. The most effective way to grow organic traffic and web based business is to provide good and seamless user experience to the user, The core web vitals metrics and threshold are defined by google which provides publishers , developers , businessmen a proper ways to make their websites , fast, responsive, interactive and optimized. In this research paper we will discuss how we can improvement the core web vital score of website using Next JS. Economic Times achieved overall 43% better bounce rate across the entire website by optimising the website performance using Google core web vitals by improving the CLS by 250% and overall LCP by 80% from October 2020 to July 2021. According to Eja Rakotoarimanana, Consultant, Fifty five , getting LCP under 1 second leads to large increase in conversion and reductions in bounce . Renault able to decrease the bounce rate by 14 % and increased the conversion

by 13 % with 1 second improvement in LCP. Since early 2020 , top 5 European markets , the number of visitors has improved by an average of 22 percent for Renault domain from 51% to 73%. According to Alexandre Perruche , Head of Performance , Renault , a deep understanding of digital performance is key to ensuring a continuous optimisation of brand site. By keeping good LCP Rakuten have achieved 61.31% increase in conversion rate , 26.09% increase in revenue per visitor , 11.26% increase in average order value. A good FID score can lead to an increase of upto 55% conversion rate , 53.37% in revenue per visitor, 33.13% in conversion rate , 15.20% in average order value , 99.9% in average time spent , 35.12% reduction in exit rate . Vodafone found that 31% improvement in LCP leads to 8% more sales , 15% uplift in the lead to visit rate, 11% uplift in the cart to visit rate . RedBus improved their websites INP which increased sale by 7% . The relationship between business health and page performance is well known , Though INP is a relatively new metric 20 compared to other core web vitals, RedBus observed better business outcome by focusing on improving that important user centric performance metric, and the result is they got 7% increase in overall sales .

2.4: Modern Web architecture or framework for website optimisation

According to Kynash, Yurii, Kolomoyets, Mykola (2023), the mostly used front-end architecture are React Js and Next JS because every project needs proper structuring and inner architecture provided by Next JS and React JS which fits perfectly with business requirement which allows developers to keep developing the projects more easily, support them, and extend them without any damage or at least with minimum risks. According to Dinku, Zerihun (2022) React.js and Next.js are among the JavaScript frameworks for building web applications.

2.5: Next JS Feature

According to Fariz M, Lazuardy S, Anggraini D (2022) Next js have predefined built in css support , The Next JS boilerplate by default is having global.css file , this file makes css globally accessible throughout the application. Next js framework is having a easy routing mechanism or support located in the page folder . So it doesn’t need any third party library to do the routing .The Next JS framework already support server side rendering in rendering webpage , so it will not make the user to see blank page in the initial load and reduced the load on browser . Next.js will load the HTML first then Javascript which makes the site

visible , then internally it works to make the site interactive by loading the JavaScript. Next JS supports pre-rendering with server side rendering and server site generation method, so it can be adjusted based on application needs . In server site generation mechanism , pre rendering method generates the HTML during the build process.

According to S.Sasikumar, S.Prabha, B.Chandra Mohan (2022) Next.js organise and bundles all the configuration files and packages together in a neat way .Tong, Jason, Jikson, Ricky Rivaldo , Gunawan, Alexander Agung Santoso (2023) analyzed DOM CRUD operation means create , read, update and delete operation and also analysed the respective time frame for each framework , the analysis showed that Next JS provided very good performance when compared to other framework, React JS does not able to achieve the speed of Next JS initially because of investing in DOM operations, where as Veu JS demonstrate average speed performance.

According to Harish A Jartarghar, Girish Rao Salanke, Ashok Kumar A.R, G.S ,Shivakumar Dalali (2022) Next JS framework optimises the performance of the website and can be used to build React apps with server side rendering , it also improves the SEO of the web application and makes website easily accessible by improving the website crawling score. React JS and Next JS builds the significant building component across the digital world , because they have the ability to improve the performance , optimize the website , reduce the development cost and provides better SEO scores.

According to Sushin Pv(2022) the major reason for utilising Next JS is to have a default support for SEO , we can use Next JS script and can use different strategy to load the required third party scripts , while developing a page we should always keep our eyes on the build size , loading properly sized image and prefetching them impacts the page speed of the site a lot , we can use NextImage , it is a lazy loader technique to load the image , it is also vital to load images which are on the first fold of the website . Like if we have 300*300 px in canvas , downloading 1200*1200 px will be waste of internet bandwidth.We can use dynamic import for lazy import which will prevent all unnecessary code from being bundled together.Next JS preloads all the JavaScript for pages having a link tag to them from current page , this speeds up the client side rendering , but if we are having a link in footer , user more likely will not click on those links , so we can mention prefetch=false on those link

tags to prevent them from been preloaded . Next Js also provides builtin support for AMP, which increases page speed by creating high value page to AMP and make them load more faster.

2.6: Summary

After reviewing published works of literature, the fact is established that performance and accessibility of websites plays an important roles on growing business and retaining users . In this research paper we will be identifying challenges and factors that can affect the website performance. We will be using google core web vitals metrics and google page speed insights tool to measure the performance of the website. The search engine ranking of huge number of website in google search engine is determined by Google. With consistent investigation in research and development of many number of tools over many years , core web vitals and Google page speed insights have become one of the widely used metrics for performance. In this paper we will also analyse how website rendering happens and how those factors are responsible for slowing down the rendering process of the website, and finally we will explore how Next JS can improve the performance and seo of the website by optimising those factors highlighted in google page speed insights reports following Google web vital standards. In this paper we will mostly focusing on the technical and practical approach of how we can implement Next JS in our application to optimise the website performance rather than discussing on the theoretical aspect which are already present to most of the published works of literature , We will also identify the scope of improvements of the application by analysing the page speed report and will address those issues using Next JS optimisation techniques

CHAPTER-3

RESEARCH METHODOLOGY

3.1 Importance of Performance

The success of online venture is highly dependent on website performance, retaining user is very important for improving the customer conversion , website that respond to the user interaction and inputs quicker and in proper way can retain and engage user more than the website which is non responsive or have a slow response time, performance can also have a material effect on whether your website will fall through, performance is not just about business status , when it comes to user experience speed matter , As a page loads there is no user experience or interaction in the period when the site is loading , at that interval user are actually waiting for the content to come on the site , this lack of experience is fleeting and facilities customer towards lower commitments since user are forced to wait . Website comes with many images , third party script and lots of code , browser needs to download those assets for rendering those assets in the site , which need few megabytes of users data plan, specially for mobile device , with limited CPU power and memory downloading those assets can create poor performance conditions. In order to give good experience to the user performance is very important aspect. When it comes to investors and stakeholder they should always value the importance of website performance , because it comes with all other aspects like user centric performance metrics, customer satisfaction which finally leads to business growth and improvement .The way to achieve a excellent performing website depends on where our website is present in current scenario from performance prospective , what is the user complexity , design complexity , what are the easy fix we can apply to the website which will show a significant impact on performance growth of the application .By delivering a good performing , responsive and quick interactive website to our users we are giving our customer a seamless delightful experience , which make them our loyal and returning customer .With the advent of framework like Next JS we can optimize our website very easily .

3.2 Critical Rendering Path

Optimisation critical rendering path is essential for improving a web application. The critical rendering path follow different steps which are as follows:

a> It first construct the Document Object Model (DOM) from the HTML received.

b> It then construct the CSS Object Model (CSSOM) from the css file it received.

c> It use to identify and apply the Javascript that can alter the document object model and CSS object model.

d> It used to construct the render tree with the DOM and CSSOM created .

e> It use to perform layout and style operation on the page to observe which element will fit in which part of the DOM.

f> It paints the pixels of the element in memory and compress the pixels if any of them are overlapping.

g> Finally it draws all the resulted pixel to the screen of the browser.

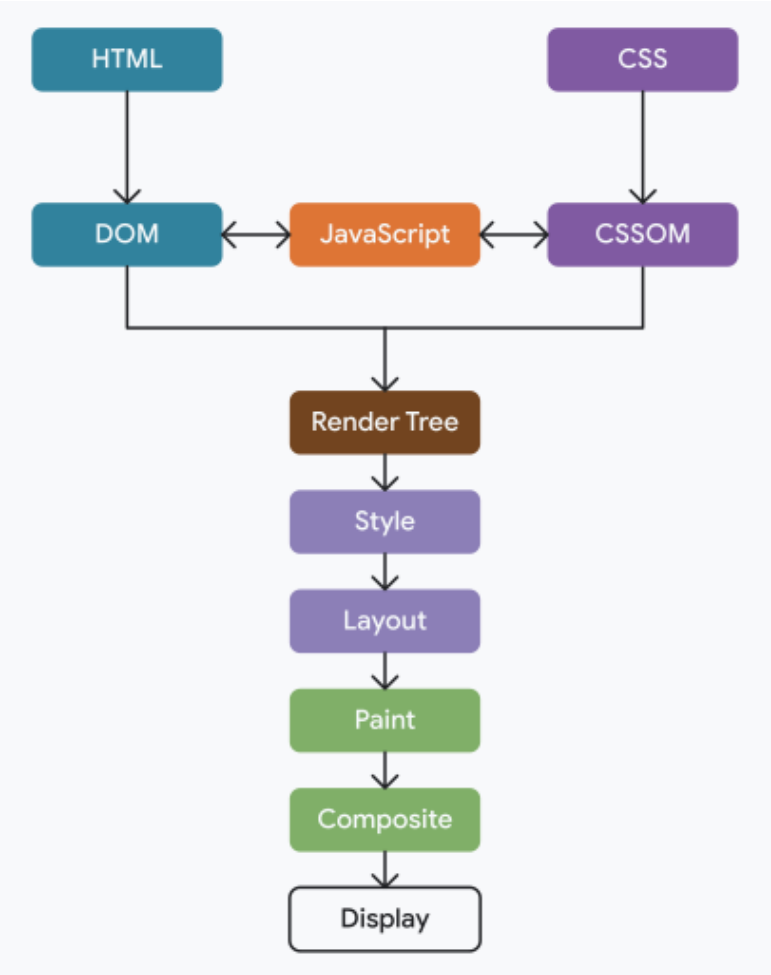


Figure 5: Critical Rendering Path (web.dev, 2023)

3.3 Advent of V8 to redefine JavaScript

Google created the Chrome V8 engine in 2008, before the introduction of V8 the main job of Javascript was to communicate with css for building a user interface and it used to execute few common function like validation etc. With the advent of chrome V8 engine google pushed JavaScript to the forefront of the HTML5 . According to Jartarghar HRao, Salanke, GKumar , ADalali (2022) V8 JavaScript actually redefines the JavaScript which increased the speed 56 times quicker than any version of Internet Explorer (IE).Traditional web browser use to create JavaScript by parsing byte code and compile whole web project to generate the code and once JS is created it will get complied from file system . As a result JavaScript execution is a big blocker and is significantly longer than any complied language like Java and C++. Because of the excellence of V8 engine, different JS platform based on V8 engine started emerging which brings a new era on website development .Node Js was introduced on 2009 , with Node js developer started understanding that JavaScript can do more than just running a simple script on a webpage . Now let's discuss two important concepts that solidifies the concept of web development and render process ,Client side rendering and Server side rendering .

3.4 Client Side Rendering VS Server Side Rendering

3.4.1 Client Side Rendering

Client side rendering uses JavaScript to render the content in browser , as a result initially few part of HTML document loads with Javascript, rather than waiting to receive all the page content directly from HTML document , the remainder of the site is then rendered using the browser .Initially the page load used to be slow with CSR but sequentially all the pages loads very fast . In Client side rendering server obtains data at run time and each call does not requires complete User interface reload , it only used to refine the DOM element with the updated data . Famous framework like React JS , Angular JS , Vue JS uses Client side rendering .

3.4.2 Server Side Rendering

In Server side rendering server send the whole HTML response to the browser over the internet , it is browser responsibility to render the page , server use to do entire process like requesting data from the database , creating the HTML and sending it to browser to render etc . Next Js uses Server Side Rendering .

3.4.3 Client Side Rendering Vs Server Side Rendering

According to Jartarghar HRao, Salanke, GKumar , ADalali (2022) the main difference between client side rendering and server side rendering is that in server side rendering server sends ready to render HTML of an application to the browser, but in client side rendering browser will just get a sparse of document with a link to Javascript.In both the situation, it will go through the step like creating a virtual DOM and adding event to make the page interactive . In server side rendering the user can see the page from beginning while all the process are happening . In client side rendering the same process is followed before the virtual DOM is sent to the browser .Web application when use server side render approach loads quickly , where as when it loads with client side rendering it shows blank, white page initially. Before going into the details about web performance , we should first understand the render process of a website , because to understand how to optimize a website we should be aware of how website rendering process happens.

3.5 Process of rendering Website

Many activities or process used to happen under the hood on the process of rendering a website . Renderer process is responsible for everything that is happening inside a browser tab . Main thread mostly have to parse most of the code we send to the server.There are two other threads called compositor and raster thread which helps to make the render process efficient and smooth. According to Mariko Kosaka (2018) the main job of renderer process is to convert JS ,CSS and HTML into a webpage that can interact with the user . Now let us go through each of the rendering process in details.

3.5.1 First step is construction of a Dom :

At first the navigation tab used to send commit message to renderer process , to start receive data of HTML , then HTML is parsed by the main thread which is defined by the HTML standards. The internal structure of a page and data of the browser by which web developer can communicate with via JS is called DOM.

3.5.2 Sub resource Loading:

A website usually have many extra resource like JS , CSS , image etc, these resource needs to be downloaded from network or cache to build the page . The main thread use to request them one by one , but in order to speed up this process preload scanner is used , preload scanner used to preload <image > and <link> tags by peeking at the token generated by HTML parser and sending request to the network thread.

3.5.3 JS can block the parsing:

When the HTML parser encounters any script tag , it used to give it more priority by pausing the whole parsing process of HTML and start executing , parsing and loading the JavaScript first because it can change the structure of the DOM . That's why the HTML parser have to parse the JS script first before it can continue parsing the HTML document .

3.5.4 Hint to browser how you want to load resources :

There is a way by which we can give hint to the browser in which order they can order the resource , we can add async or defer attribute to the script tag , and give hint to the browser to load and run script asynchronously , without blocking the parsing. We can also use <link rel="preload"> tag to inform the browser that the resource is definitely needed for website navigation , and it needs to be loaded as priority.

3.5.5 Style calculation:

The main thread also parse the CSS and determines the computed style of each DOM node , it used to define what style will get applied to each of the element of the DOM structure based on CSS selector. Each DOM have a computed styles like <h1> is bigger than <h2> etc. We can check the computed style of the components of a website in the computed style tab in dev tool.

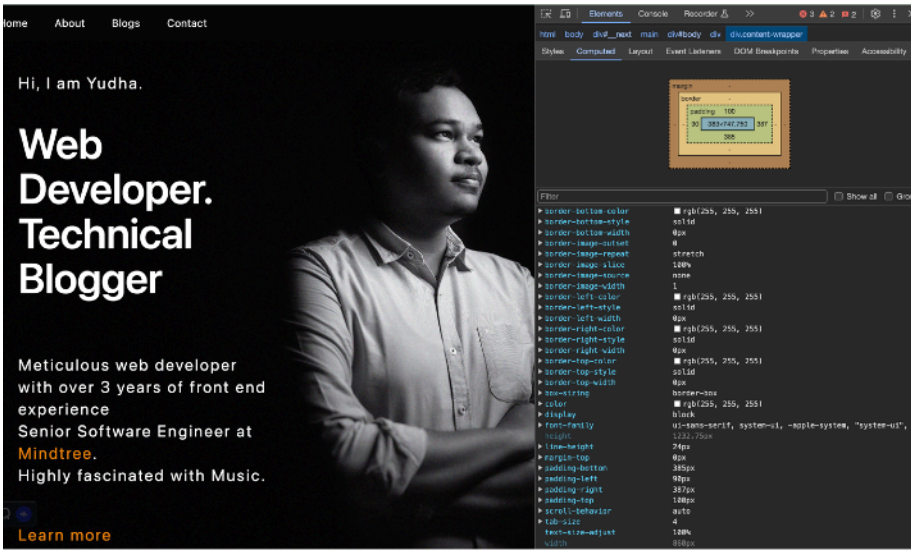


Figure 6: Computed styles of the application

Since we have created the DOM and computed style , now let's create the geometry of the elements to build page .

3.5.6 Layout :

Layout process is the process of designing the geometry of elements, Layout is similar to DOM tree , only Layout contains the data of the visible component on the page .Main thread creates the layout which will have information about the x and y co-ordinate of the element with boundary box size by crawling through computed style and DOM .

3.5.7 Painting :

So now we are having the DOM , computed style and layout of the website , in order to reproduce the painting of the website we need the location and order of the element. One interesting point of renderer process is in each step the result of the previous operation is used to create new update , so if there is any essential change in the layout tree , the past order also needs to be regenerated for the affected part of the document . Usually browser

use to run this operation in every frame (60 times in a second). Even if the renderer operator keeps reloading the document and refresh it in every frame. The calculation happens in the main thread only, which may block the application when Javascript is executed. For this reason it used to cluster the JavaScript into smaller chunk and schedule those to run at each frame. Now the browser knows the structure of the document , the style of each element and the proper ordering of the website elements. Turning these element into pixel on a screen is called rasterising. Previously chrome used to raster when user scrolls the page . Modern web browser used a sophisticated concept called compositing , which separates the part of the page into layers, then raster them separately and composite a page in a separate thread called composition thread.

3.5.8 Dividing the layer:

Main thread used to walk through the layout to create layer and try to find out which element needs to be in which layer.

3.5.9 Raster a composition of the master thread:

Once the layer is created and proper order are determined . The main thread sends those details to composition thread , after receiving the details composition thread starts rasterising each layer . The composition thread used to prioritise differ raster thread so that the elements which are coming within the viewport rastered first . Once tiles are rastered, composition thread used to gather tile information and store in draw quad or composition frame .

Draw Quad: It contains information such as the tile's location in memory and when is the page to draw the tile taking in consideration of the page compositing.

Composition frame: A collection of draw quad represents a frame of reference of a page .

A composition frame is then submitted to the browser processor via IPC. And then composition frame are stored in GPU to display it on a screen.The benefit of composition is that it is done without involving the main thread, composition thread does not need to wait for the style calculation or JavaScript execution.

3.6 Factors which affects the web performance :

Web performance is a very critical aspect of web development, by optimising our website we are giving our user a better experience which help us in increasing traffics of our website and finally leads us to business improvement. Keeping this in mind Google have initiated Core Web vitals to provide unified guidance for quality signals that are essential for delivering a great user experience on the web. In the current scenario consumers increasingly rely the web to access digital content and service, while they are browsing through your website they are actually comparing you and rating you with your competitors against the best-in-class service they use every day .So overall we can say performance is a foundational aspect of good user experiences. We will also try to find out what are the factors and features which can affect the page speed of website. Native web-app typically have a initial phase and running phase , however unlike native applications for web page and web application the lines between these two phase are much less . Once browser have resource to render a page the browser decides when to render and when it is too early, if browser render as soon as it just have some components with css and js , the page can momentarily look broken and on the other hand if browser wait for all new resources to be available instead of doing parallel rendering the user will be left waiting for long time , so sequence of steps the browser takes before performing initial render is very critical and is know as critical rendering path , when it comes to webpage performance improvement and optimisation minimising critical rendering path plays a vital role.

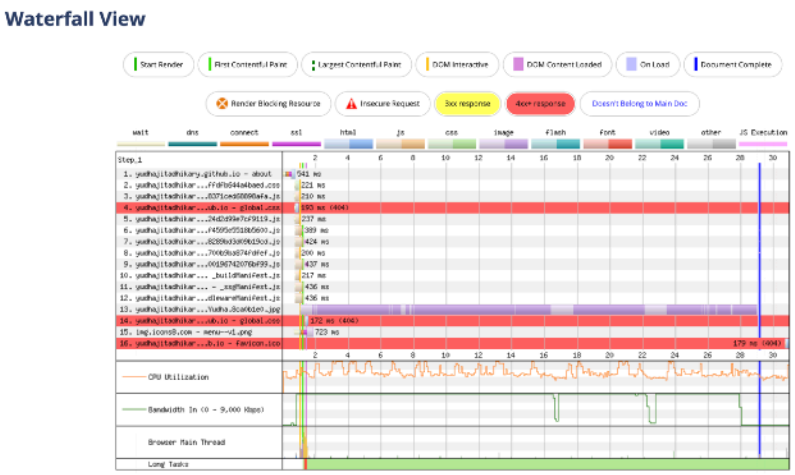


Figure 7: Waterfall view of https://yudhajitadhikary.github.io/Portfolio/about

Some resource are very critical for which browser pauses entire rendering until it dealt with them. Browser try to be as efficient as possible whenever it sees css resources it pauses rendering and will start processing rest of the HTML ,so it will block rendering , so such resource (mainly css) are know as render blocking resource .Another resources are parser blocking resources these are the resources that prevent the browser from looking for other work by continuing parsing the HTML,Mainly Javascript by default is considered as parser blocking resources. In this paper we will be analysing how we can optimize those resource using Next JS strategies and we will discuss how Next js , a popular React framework , can help us improving the performance of our website.

According to Bangzhong Cao ,Minyong Shi , Chunfang Li (2017) the steps followed to render a webpage are as follows :

1> When user enter the access url in the address bar of the browser.

2> The browser checks for the IP address in the domain name resolution library.

3> The client browser sends the request url to the server.

4> When server receives request to generate HTML from the browser , the server then sends response to the browser request.

5> Browser accepts a parsed data send by the server , server also returns the related resource file to the browser.

6> Finally browser will be parsing the related resource file and render the page from top to bottom.

Now let's try to figure out the waiting time involved in this whole process:

1> The time when server are resolving domain for the server IP also know as domain resolution time .

2> Time required by a client to establish the connection with the server.

3> Time taken by the TCP to establish a connection .

4> Time server takes to respond to the HTTP request.

5> Time required to download the request content after the server respond to the request.

So the most common factor to slow down page speed on a web page includes scripts, fonts , plugins (html, Javascript and css).

3.7 Current trends of Next JS

In 2024, Next.js development trends include hybrid rendering for optimal performance, a focus on accessibility and inclusivity, and enhancements to the developer experience with features like built-in TypeScript support and advanced debugging capabilities. To verify the current popularity of Next JS , we will go through few surveys recently done on few well known websites and platforms .

3.7.1 Google trends:

To understand the recent status of popularity of Next Js, we can check the google trends portal , google trends is a portal where we can compare the interest over time graph of several keyword or topics . When we compare Next JS with few other frontend development packages like vue.js , nuxt.js .We can see that Next js is having a very consistent and highest interest/search index , than other two frontend packages .

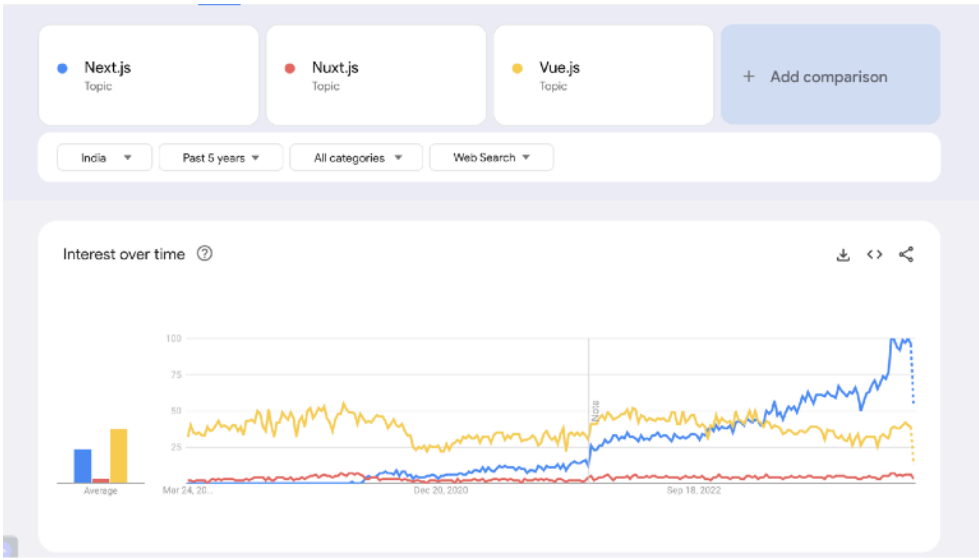


Figure 8: Interest over time on Google trends

3.7.2 Npm trends:

Npm trends is website to compare packages download statistics , GitHub stars , ratings etc . When we compare our frontend packages (nuxt js and vue js) with Next JS with respect to number of downloads in past 1 year. We can see that Next js is having the highest number of download in past 1 year.

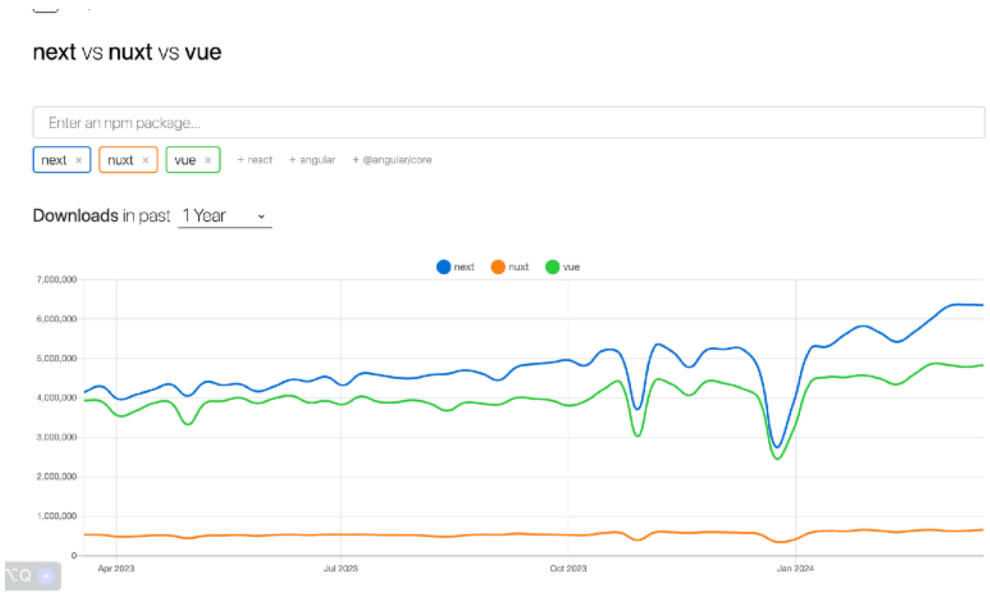


Figure 9: Npm download record in past 1 year for Next JS

3.7.3 Stack Overflow:

Stack Overflow is the largest, most trusted online community for developers to learn, share their programming knowledge, and build their careers. We will check survey result of the most popular web framework and technologies, the respondents of these survey include of coding learners , professional developers , architects etc . So according to the report Node.js and React.JS are the two most common web technologies used by all respondents. But point to highlight from these survey is Next.js moved from 11th place in 2022 to 6th this year mostly driven by it’s popularity and optimisation strategies.

Figure 10: Most Popular technologies or framework

Now we will check how web frameworks have trended over time based on the use of their tags since 2008 in stack overflow to analyse overall trends of frontend frameworks , there also we can see React JS is highest in recent trending and as time passed Next Js over performed and become second highest trends in current years .

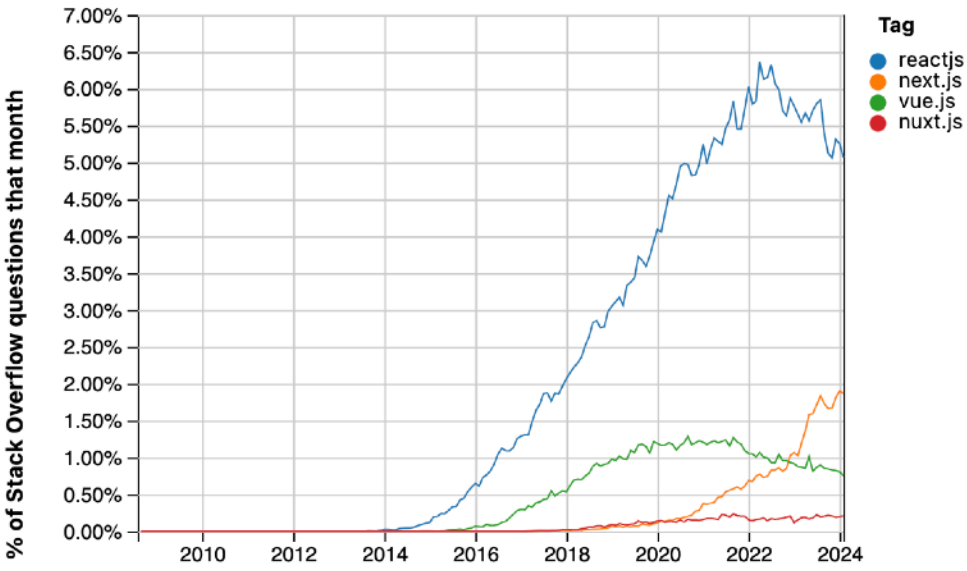


Figure 11: Overall trends of frontend framework

So we can seeing a good amount of growth on the usage of Next Js in current years , because Next js incorporates developer experience with production ready features like , server side rendering , typescript capability , smart bundling , route prefetching feature , optimised format of image , lazy loading , dynamic routing , file based routing and dynamic import etc.

3.8 Optimisation features of Next JS

Next JS provides many builtin features like file based redirection and routing, server side rendering (SSR), static side generation (SSG) which ensure improvement in performance of your application.Next JS also provide functionality like automatic static optimisation, image optimisation, code splitting , font optimisation ,optimising the dependency script and prefetching.

3.8.1. Automatic Static Optimisation:

Next JS also render our application automatically as static html , if our application is not having dynamic routes. If neither getInitialProps nor ServerSideProps are defined , Next JS will not re render the page for every request , this improves the performance by decreasing static html generating time of the application . If page are having dynamic routes using Next js getServerSideProps and getInitialProps we can pre-render the pages on the server and fetch the necessary data before sending the HTML to the client , which leads to improvement in performance of the page .

3.8.2. Code Splitting:

The process of segregating our code into smaller chunks is called code splitting. Next JS automatically segregate our code into maintainable chunks so that it takes less time to load. We can use dynamic rendering of the component provided by Next JS which will load component whenever it comes on screen, this will help us to further utilise the benefits of code segregation.

3.8.3. Image Optimisation:

Next JS is having a builtin feature called next/image component which automatically resizing or compression image based on different device size. The process of loading image whenever needed is called Lazy loading , this will load image when they are visible on the screen . Automatic image optimisation helps to reduce the time to load the image in our application . Using Next/image we can load immediately the image which are on the 1st fold by setting the priority property on the next/image as true. Next JS will prioritise loading the essential images we want to load first .

3.8.4. Font Optimisation:

Next JS also use the next/font component to optimise fonts, it will automatically download the fonts on server which will reduce the load time the fonts in our application.

3.8.5. Prefetching Facility:

Next JS is having Link component from next/link, the best feature of next/link is it automatically prefetch the code of the page which user may like to visit in next stage.

3.8.6. Optimising Dependency Script:

Next Js is also having next/script component which allows browser to first load the critical component , then the script . We can also ensure the script will run once , which is highly useful in the scenario where a common script is required to be loaded all across the site .

3.8.7. Critical Rendering Path Optimisation :

Next Js also helps to reduce the critical rendering path, it focus on the process for initial rendering of our webpage In this whole process browser needs to wait for some critical and essential resources to download needs before it can start with initial render ,those are part of the HTML , render block css and JavaScript script in the head element.There are few ways by which we can improve LCP of our application some of them are minimising render blocking resource like css and js , minimising the main thread work . Application main thread parses the HTML , executes JavaScript ,render the pages , so we should target minimising the work assigned to the main thread that will improve the LCP of the application , Javascript is single threaded language , which means at a time one task can be executed , so if we target reduce the JS executed on main ,thread we will get a good amount of impact on initial load time. We can use async function and Next Js also provided a component called next/script to load script , which allows browser to download and execute script asynchronously without waiting for it to execute and blocking the render process of the page . In order to increase build performance of our application we can target to improve the network performance.

3.8.8. Caching Facility:

Using Next JS we can set the cache control header of our application by setting header property in file named next.config.js .Content delivery network (CDN) is a network of server distributed world wide where we can store static assets in the server which gets delivered to user based on their IP region Next JS also provides built in support of Vercel Edge Network as a CDN automatically.

3.8.9. Bundle Analysis and Tree Shaking Techniques:

Bundle analysing is an approach for analysing our application to identify opportunity for decreasing it's size. Some tools for analysing the size of our application includes next-bundle analyser , webpack-bundle analyser etc. Tree shaking is the process of removing non essential, unused and dead code from the deployable bundle, we can use ES6 modules instead of common JS to take advantage of the Next JS builtin support for tree shaking . We can use import to import statement instead of requiring a module export statement. Next Js also helps us to exclude non essential folder or files from the bundling and build process , by this we can reduce the bundling time that improves overall performance by excluding specific non essential directories, we can use exclude parameter in the filename called next.config.js to exclude specific directory or file , by excluding non essential resource from deployment bundle we will decrease the amount bandwidth of data that needs to be processed , which leads to faster build times and efficient resource utilisation.

3.8.10. Server Side Rendering :

The Next JS framework already support server side rendering in rendering webpage ,so it will not make the user to see blank page in the initial load and reduced the load on browser. Next.js will load the HTML first then Javascript which makes the site visible , then internally it works to make the site interactive by loading the JavaScript. Next JS supports pre-rendering with server side rendering and server site generation method, so it can be adjusted based on application needs . In server site generation mechanism , pre rendering method generates the HTML during the build process.

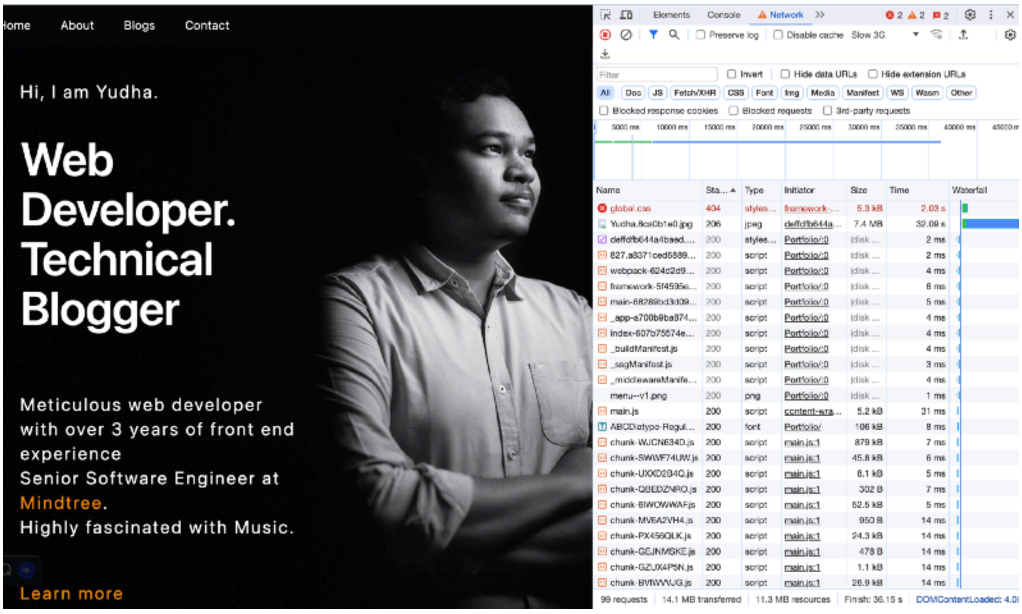


Figure 12: Next JS loading styles before Javascript

3.8.11. Easy Routing Feature:

Next js framework is having a easy routing mechanism or support located in the page folder. So it doesn't need a third party library to do the routing .

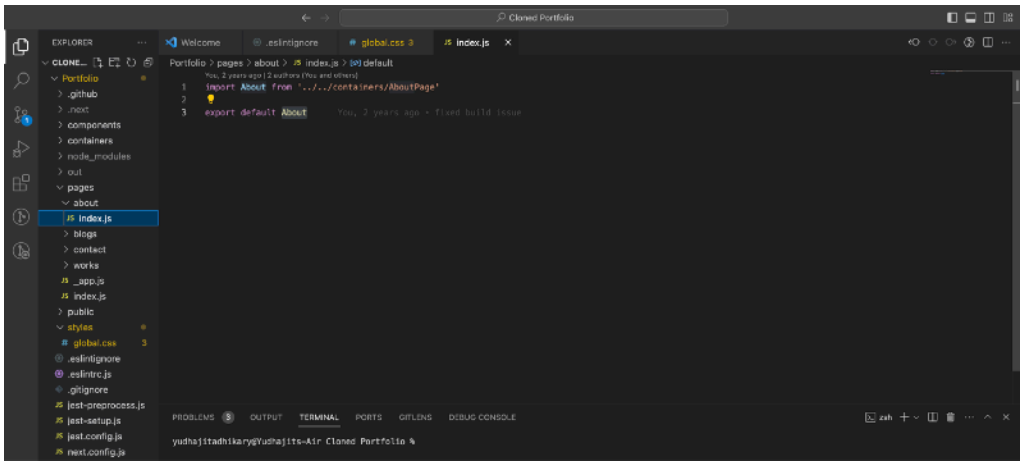


Figure 13: Page folder for routing mechanism in Next JS

3.8.12. SEO Benefits:

Next JS provides various data fetching mechanism , because it supports data fetching mechanism using client side rendering , server side rendering , server site generation and incremental static regeneration. Next JS has automatic code splitting , a technique of separating the application bundle into smaller chunk, this technic are known as code-splitting to reduce the initial load time of the application. Next JS supports dynamic imports , some component are imported during the run time and other components will get imported whenever it's necessary, this concept is known as lazy loading. This deferred landing helps to improve the performance of the website by decreasing the amount of Javascript.Next JS uses the server side rendering , which converts the react code to html on the initial request. A single page application transmit a large Javascript to the browser , majority of the HTML will be generated by the Javascript. When SEO google bots crawls the website, examining the HTML of each url they came across, the SEO bots do not composite Javascript and they do not always want it to be converted into HTML ,since bot determines what appears on the 1st page of search result it sometimes unable to read the page , So Google will not suggest or consider it as a best option to answer to web search .Next JS helps us the encapsulate the single page application which retains the SEO benefits of the traditional website architecture .

3.8.13. Key Optimisation Features of Next JS:

According to Dinku, Zerihun (2022) Next js incorporates developer experience with production ready features like , server side rendering , typescript capability , smart bundling , route prefetching ,optimised format of image , lazy loading , dynamic routing , file based routing and dynamic import etc. According to Desamsetti, Harshith ,Dekkati, Sreekanth(2021) Next Js helps to optimize the website performance by focusing on the server components, simplifying the process of obtaining data, making applications using new page directory, online streaming and internal tooling. According to Fariz M, Lazuardy S, Anggraini D (2022) Next JS already has built-in css support , in Next JS boilerplate we are already having a built-in css support , At the beginning of the installation there is a global.css file and modules , this global.css is a css file in the Next Js framework that is globally accessible throughout the application.

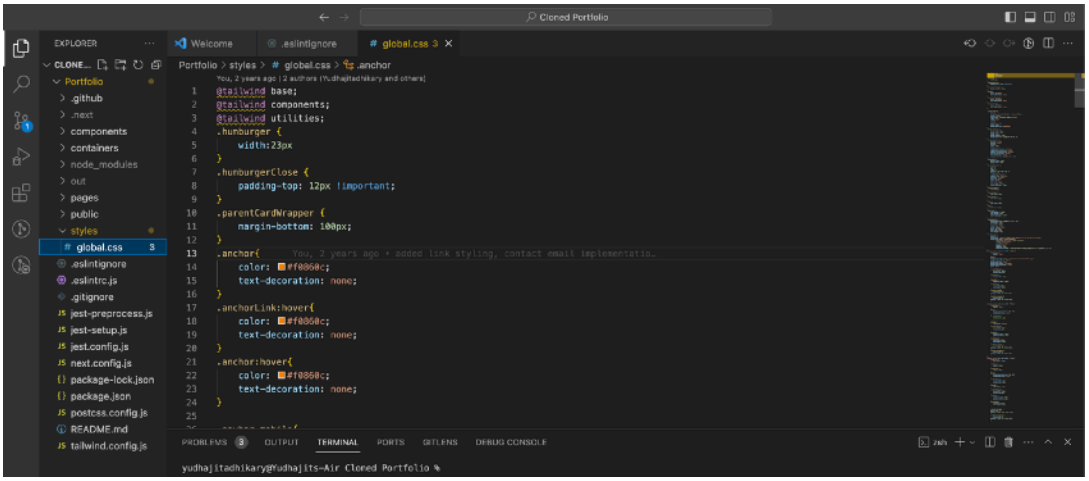


Figure 14: Built-in global css file in Next JS

Now let's summarise the key optimising features of Next JS in tabular format .

Optimisation	Description
Incremental Static Rendering	Next JS provides incremental static rendering concept which reduce the user delays without overwhelming the site with multiple resources , incremental static rendering generates HTML file and used to refresh or regenerate the HTML file again after certain amount of time rather than only once like static site generator for in every request like server side rendering .
Reduction of Javascript Bundle Size	Next JS reduces the Javascript bundle size on initial load by using code splitting and react lazy load and suspense component which loads necessary components or modules when needs or when user come to that exact fold, which leads to faster page load time and improved user experience.
Internalisation support with next-i18next	Next JS allows us to develop language and country specific pages which provides ability to change or switch between them using dropdown having language menu , this improves the user experience of the entire website.

Optimisation	Description
Optimisation of image with next optimize images	Next JS allows the optimisation of image of the site , rendering the overall page load time on improving the user experience , as well as improving the SEO by reducing the size of the page .

Table 1: Key optimisation features of Next JS

3.8.14. Best Practices for using Next JS:

Next.js is versatile and can benefit various industries, including e-commerce, healthcare, finance, education, and entertainment. Its scalability, performance, and flexibility make it suitable for a wide range of applications across different sectors. Let 's see the best practices we should follow to create a robust , scalable , flexible and optimized website using Next JS.

Best Practice	Description
Use NextJS for SSR	Next JS allows server side rendering which reduces the initial loading time by allowing content updates without delay, which can improve the performance and Seo score of the website .
Implement code splitting using React Lazy load	Next JS can reduce the size of JavaScript bundle by using lazy loading and suspense feature which improves the performance and user experience of the website .
Optimising image using Next image optimisation	Next JS is having image optimisation feature which helps to optimise the size of the website , reducing the page load time and improves the user experience and SEO.
Implement international support using next-i18next	Next JS allows us to develop language and country specific pages which provides ability to change or switch between them using dropdown having language menu , this improves the user experience of the entire website.
Use incremental static regeneration of content update	Next JS allows content updates to be reflected on the site without overwhelming resource which reduces initial load time and improves the user experience.

Table 2: Best Practices for developing Optimised application using Next JS

3.8.15. Benefits of Next JS:

Next JS provides solution for developing an performance optimised website . A optimised , seamless , well performing website will give a good user experience and will always ensure good SEO score of the site . According to Patel, Vishal (2023) site having good SEO score will be recommended by google first whenever user searches for any related keyword or queries which will increase the number of organic traffics of the site. Let's summarise what business impact Next JS can make by building a performance optimised website .

Benefit	Description
Improved performance and SEO	Implementation of Next JS comes with several optimisation techniques which significantly improve the website performance or SEO which includes faster page load time.
Enhanced user experience	Optimisation techniques such as internalisation support , reduced JS bundle size improves the user experience by improving page load time which gives smother , more seamless browser experience.
Increased organic traffic and revenue	Website having improved SEO and performance helps to increase the organic traffic as it provides good experience so users are more likely to visit or engage with the website.

Table 3: Benefits of developing Optimised application using Next JS

3.9. Real World Application

In this paper , we will be building a real world application to evaluate the performance impact of Next JS .We will be developing a static developer portfolio site using React JS and Next JS. The application will have four following pages .

3.9.1. HomePage:

It contains the short introduction of the developer.

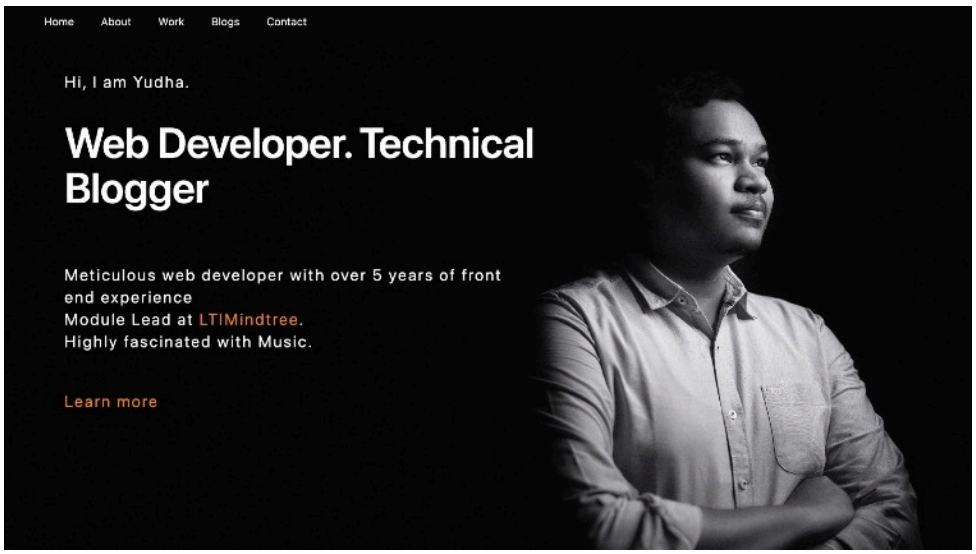


Figure 15: Homepage of Application

3.9.2. AboutPage:

It will have the detailed description of tech stack of the developer.

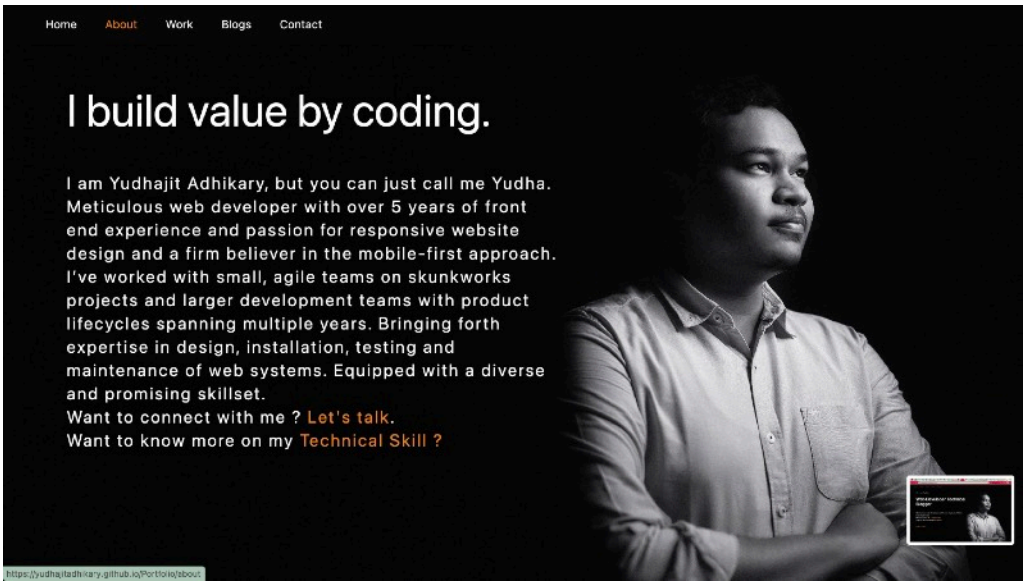


Figure 16: AboutPage of Application

3.9.3. BlogPage:

It will have the list of all technical blogs written by the developer , the blogs are linked to actual blog post so that user can navigate and read the blogs.

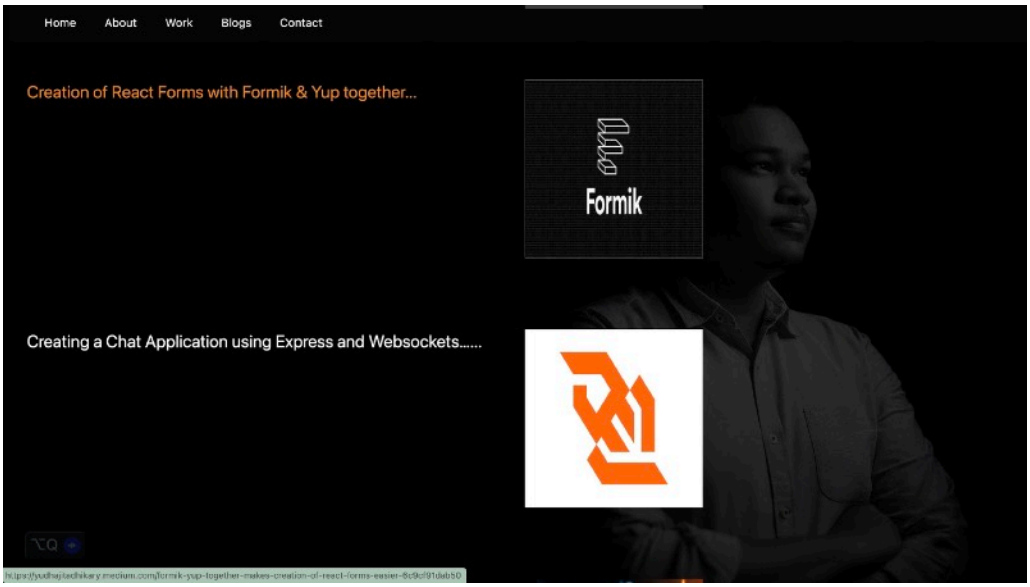


Figure 17: BlogPage of Application

3.9.4. ContactPage:

It will have developer's contact details, and anyone can directly reach out to the developer via email by filling the form provided in the contact page.Whenever a user fills the form a mail will be triggered to developer with all the queries and details filled in the form .

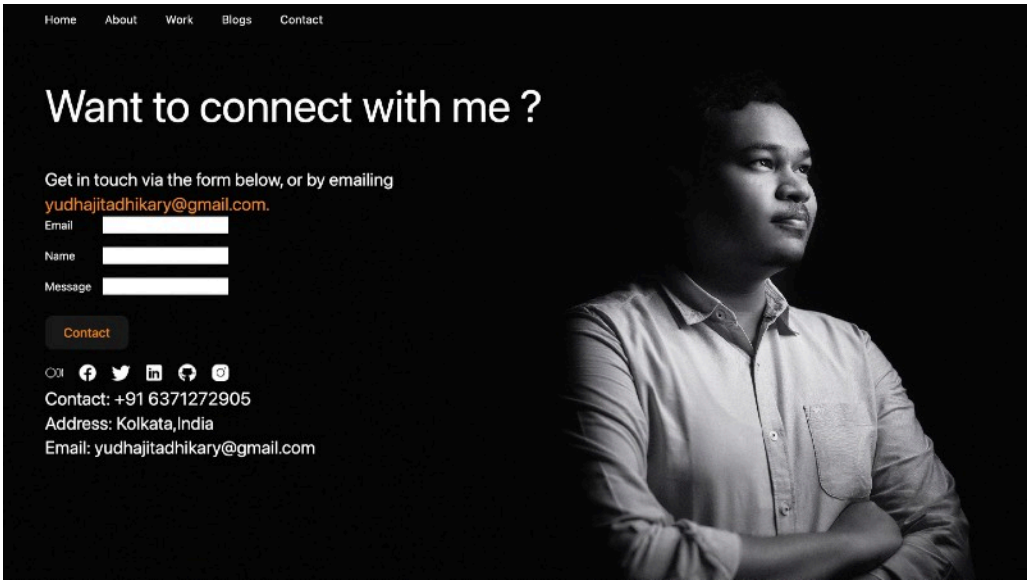


Figure 18: ContactPage of Application

3.10 Building with React JS :

React JS is having it's own boilerplate which makes easier for developer to start with coding. So we will first run the command **npx create-react-app <project name>** , this will create the boilerplate of react application, which looks like this.

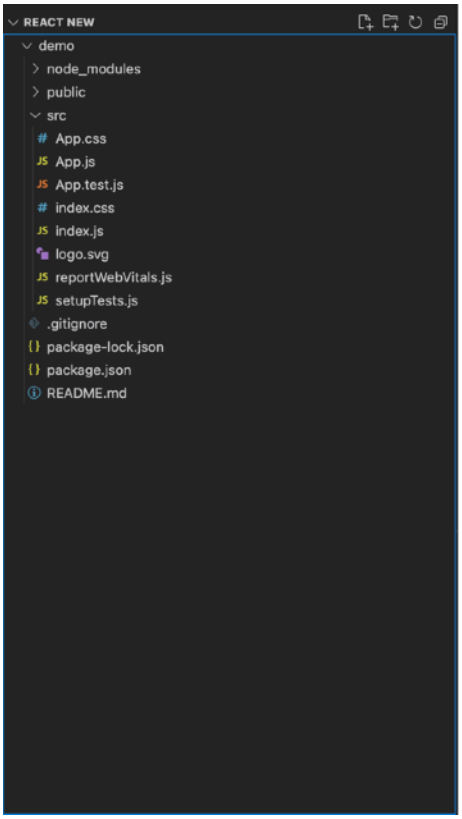


Figure 19: Folder Structure of React Application

So the boilerplate codebase is having **package.json** which consist of all the dependency npm packages we are using on our application, When we run **npm i** to install the packages mentioned in the **package.json** , 2 files will get generated , **package-lock.json** and **node_modules**, **package-lock.json** describes the exact npm dependency tree that is generated , **node_modules** acts as a cache for the external modules on which our project is dependent , it stores the npm package which are downloaded from web, **public** folder will store all the static contents and assets we are using in our application.

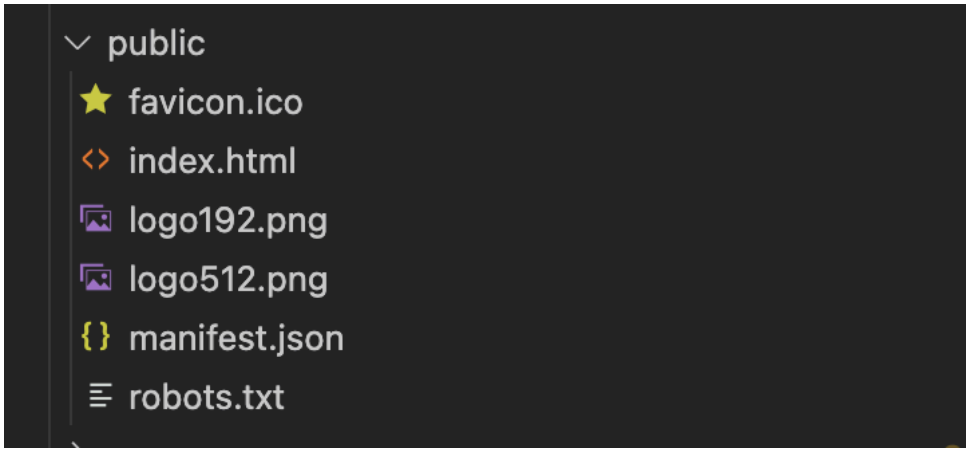


Figure 20: Public folder of React Application

README.md explains the details of the project , src folder contains the actual code of our application. Our code starts from index.js where we are importing **App.js** , it contains the actual HTML code that we will be seeing when we run the code in our local by running the command **npm start** .

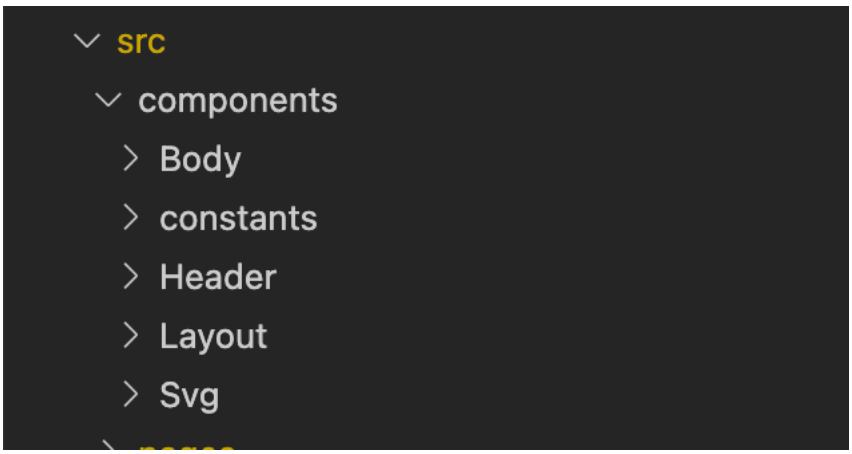


Figure 21: Components folder of React Application

We have added components folder inside **src** which contains all the components of our application .

We have also added **pages** folder which will contain all the pages of our application.

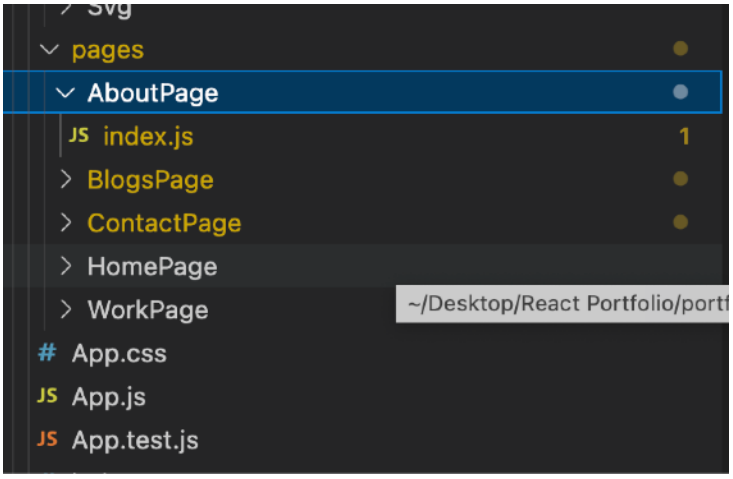
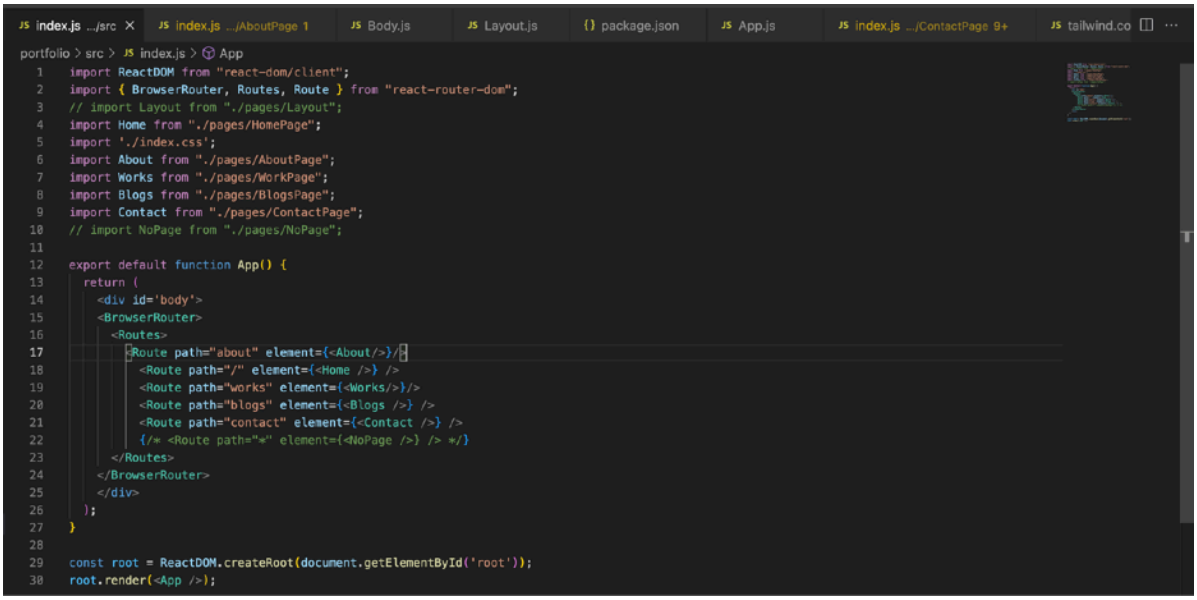


Figure 22: Pages folder of React Application

Before going to **index.js**, we should also consider that react does not support routing by itself , so we have to use npm package called **react-router-dom** to handle routing .In index.js we are importing all the pages of our application.Then we are defining our App component where we are mapping those pages with the respective routes using **react-router-dom**. Finally we are creating a dom having id “root” using React DOM and inside that DOM we are rendering our App component .



```
import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from "react-router-dom";
// import Layout from "./pages/Layout";
import Home from "./pages/HomePage";
import './index.css';
import About from "./pages/AboutPage";
import Works from "./pages/WorkPage";
import Blogs from "./pages/BlogsPage";
import Contact from "./pages/ContactPage";
// import NoPage from "./pages/NoPage";

export default function App() {
  return (
    <div id='body'>
    <BrowserRouter>
      <Routes>
        <Route path="about" element={<About/>}/>
        <Route path="/" element={<Home />} />
        <Route path="works" element={<Works/>}/>
        <Route path="blogs" element={<Blogs />} />
        <Route path="contact" element={<Contact />} />
        {/* <Route path="*" element={<NoPage />} /> */}
      </Routes>
    </BrowserRouter>
    </div>
  );
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App />);
```

Figure 23: index.js of React Application

We are using tailwind css for styling , so we have added **tailwind.config.js** file to define some basic styling related configuration for tailwind .

Figure 24: tailwind.config.js file of React Application

3.11 Building with Next JS

Now we will be developing the same application using Next JS , Next JS is build on top of React JS only , so like React JS , Next JS is also having a boilerplate available.So in order to get Next JS boilerplate installed in our system we will run **npx create-next- app@< version of Next>**. The Next JS boilerplate looks like this:

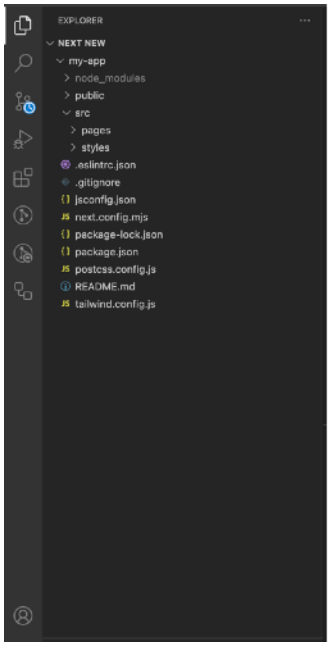


Figure 25: Basic boilerplate of Next JS

Next JS boilerplate is almost similar to React JS boilerplate , it is more segregated and atomic than react boilerplate, Next JS boilerplate is having **tailwind.config.js** by default, **postcss.config.js** is a JS file that transform our css code into an abstract syntax tree . It supports linting , minifying and injecting vendor prefixes. It produces fast build time compared to other processor . All the custom configuration of postcss can be added on **postcss.config.js** file.

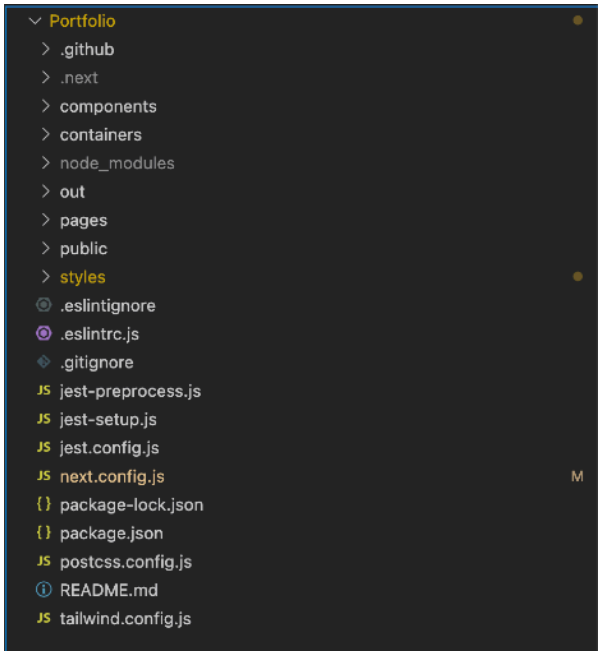


Figure 26: Folder structure of Next application

Next.config.js is a file which is used to configure Next JS configuration of our application. So to utilise the optimising feature of Next JS we will be using next/image instead of normal <img> tag , next/image is the extension of HTML <img> element with feature of automatic image optimisation, it also serves correctly sized image for each device, prevent from layout shift, it also helps in lazy loading the images , by loading images only when user enters to the viewport, So to use an image in next/image we have to whitelist the domain of the image for that we have to pass the image hostname or domain in the domains array on nextjs.config.js .

Figure 27: next.config.js of Next application

style folder contains all the css files , **global.css** is the default css file for Next JS , public folder will contain all the static assets of the application.

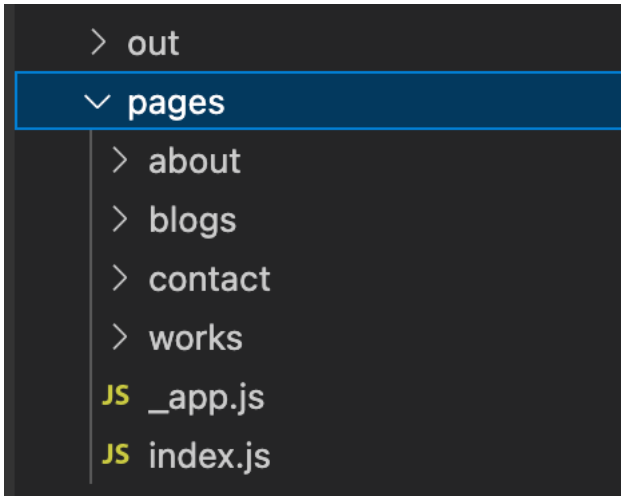


Figure 28: pages folder of Next application

Next JS is having pages folder by default , pages folder will have 3 files index.js , **_app.js** and **document.js** , **document.js** acts as the header of the application , where we can add all the third party script and tags required for our application , we can utilise next/script instead of normal script tag for adding the tags and script for our application . Next JS supports file based dynamic routing , for example in our case the code present in index.js inside pages folder will render on the base route of our application and will be considered as homepage of our application , similarly we have added the about pages as well inside pages folder , where code present in about/index.js will render on /about route and will be considered as AboutPage of our application .

```
import '../styles/global.css'
import React from 'react'
import dynamic from 'next/dynamic'
import PropTypes from 'prop-types'
const Layout=dynamic(()=>import('../components/Layout/Layout'))
You, 2 months ago | 2 authors (Yudhajitadhikary and others)
function MyApp({ Component, pageProps }) {

    return( <>
    <head>
    {/* <link rel='stylesheet' type='text/css' href='../styles/global.css'/> */}
    <link rel="preload" as="image" href="Yudha.avif"/>
    </head>
        <Layout>
            <div id='body'>
            <Component {...pageProps}/>
            </div>
        </Layout>
    </>)
}
MyApp.propTypes={
    Component:PropTypes.object,
    pageProps:PropTypes.object
}

export default MyApp

```

Figure 29: _app.js of Next application

Our code starts from **_app.js** , **_app.js** acts as the root directory of our application. We will be wrapping our application with Layout component , Layout component will be common components of our application across the page which will consist of header component which is common across the site , We will be passing the page content of our application as the children of Layout component .

```
Portfolio > components > Layout > JS Layout.js > ...
You, 2 months ago | 1 author (You)
import React from 'react'
import dynamic from 'next/dynamic'
import PropTypes from 'prop-types'
const Header=dynamic(()=>import('../Header/Header'))

const Layout=({children})=>{
    return(
        <React.Fragment>
            <Header/>
            <main>
                {children}
            </main>
            {/* <Footer/> */}
        </React.Fragment>
    )
}
Layout.propTypes={
    children:PropTypes.any.isRequired
}
export default Layout
```

Figure 30: Layout component of Next application

Now in Layout.js we are importing header dynamically using **next/dynamic** , it helps to import the component only when user scroll to the viewport , we are importing all the components using next/dynamic throughout the application, if the component is imported using next/dynamic the component will not be included in the page's initial JavaScript

bundle. The page will render the Suspense first, followed by the component when the boundary is resolved. Instead of using anchor tag we are using next/link. Next/link is extend of HTML <a> tag to provide prefetching and client side navigation between routes. **Next/link** use to prefetch when a <link> component enters within the viewport. Next JS prefetch and load the linked routes and data in background to improve the performance of the website. We are also using next/image instead of normal image tag where we can pass explicit height , width , quality and priority as well , like if the image is in first fold we can keep priority as true and if the image is in last fold we can keep priority as false .

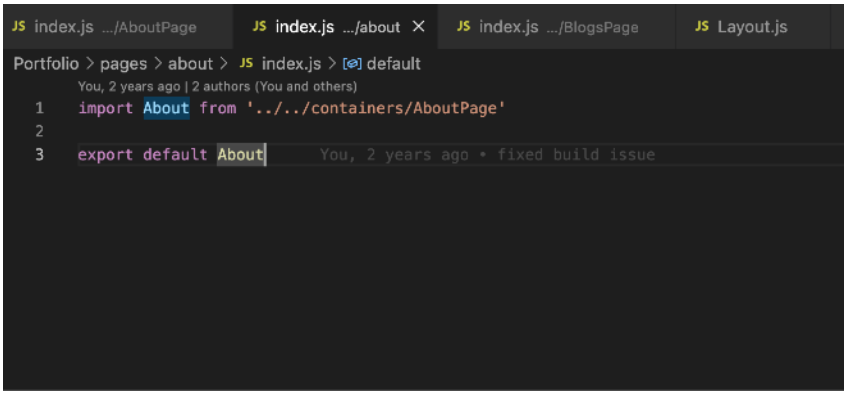


Figure 31: AboutPage of Next application

So our codebase is having pages folder , for example if we consider about/index.js file inside pages folder , we are importing AboutPage from container which is having the actual code of AboutPage .

Figure 32: Index.js of AboutPage of Next application

Container folder will have the actual code of the page. Once the coding is done we can run the code in our local. Before running the local we have to run **npm run build** to create deployable chunks of our application , the deployable chunks will be stored in **out** folder and **.next** folder will be created to store page cache and some feature to speed up our application. Once build is done we can run npm run start to run the application in our local.

3.12. Deploying with Github pages

Now we will be hosting both the react version and next version of the application using Github pages . Github pages allows us to host a webpage from our repository. First we have to signup into Github. The Github will be creating a domain in this format <Github Username> followed by .github.io, something like **<Github Username>.github.io** .So first we signed up on Github , created two empty repository with name react-portfolio for React version and Portfolio for Next version of our application. On those repository we will push the entire code of our application. In order to publish our webpage on the Github pages we installed npm package called **gh-pages**, and we added two scripts on the package.json of our application .

“predeploy": "react-scripts build” (for React application)

"deploy": "gh-pages -d build” (for React application)

“predeploy”: “next build” (for Next application)

“deploy”: “gh-pages -d build” (for Next application)

We will be deploying our application by running **npm run deploy** script. It will first run **predeploy** script which will create deployable chunks of the application then it will be running **gh-pages -d build** which will deploy the code using gh-pages.

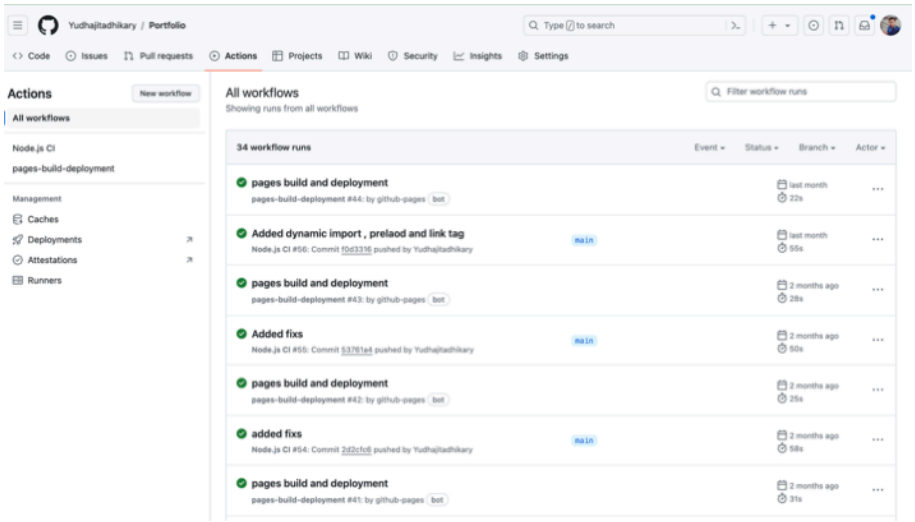


Figure 33: All build pipeline of Github pages

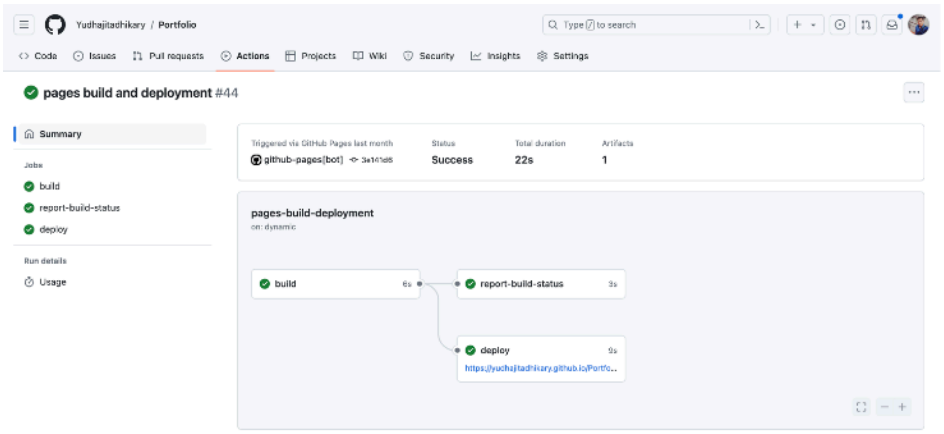


Figure 34: Build and deployment Slot of Github pages

So in this way we have deployed React JS version and Next JS version of our application on **https://yudhajitadhikary.github.io/react-portfolio/** and **https// yudhajitadhikary.github.io/Portfolio** domain respectively .

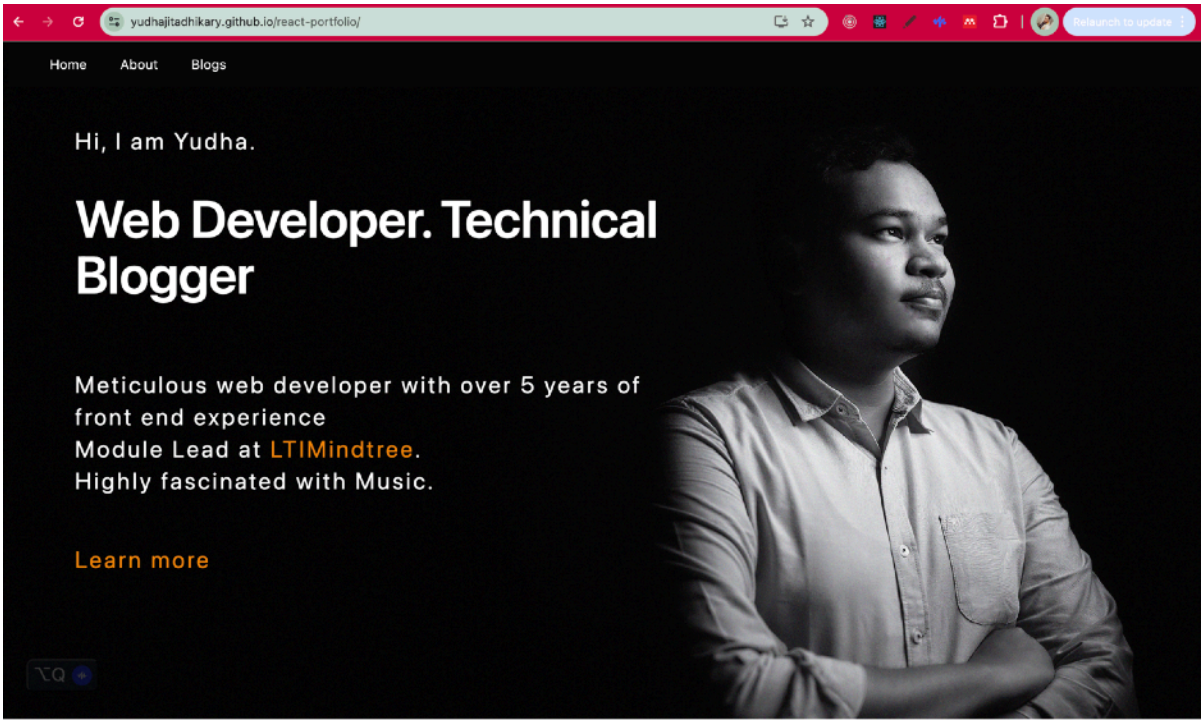


Figure 35: React JS Application

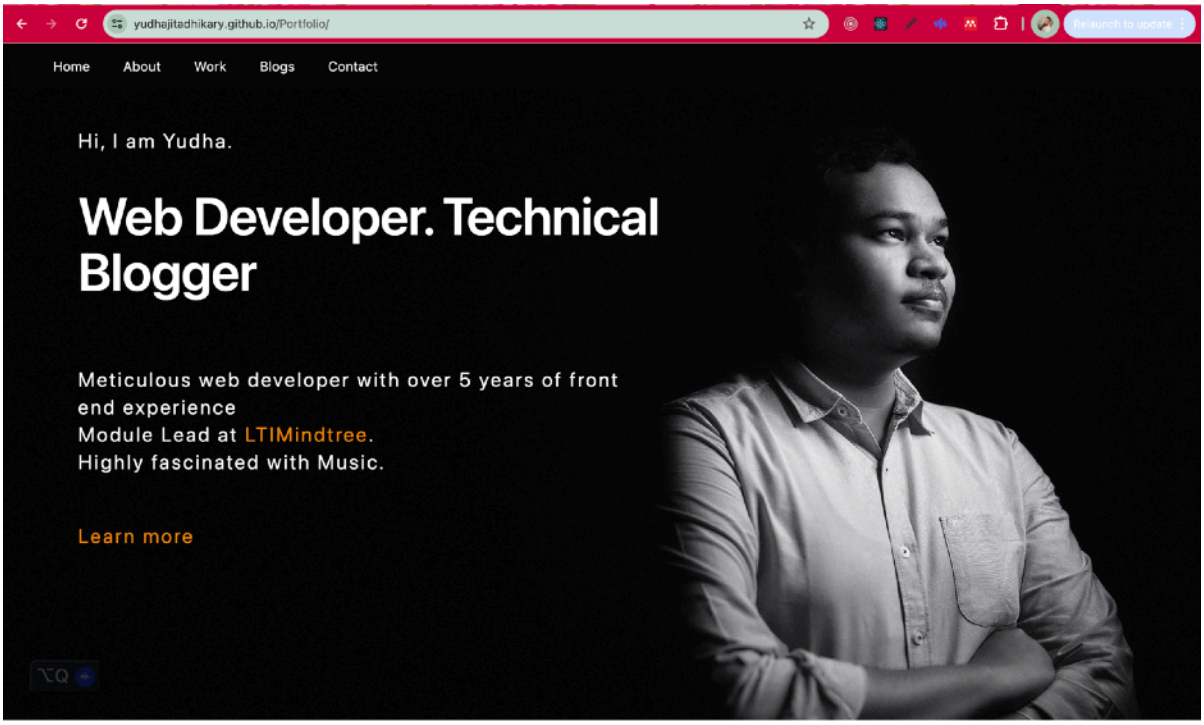


Figure 36: Next JS Application

3.13. Data Analysis Procedure:

Now we have both version of the application build and deployed on domains, To evaluate the impact of Next JS on page speed improvement, we will be running page speed scan using google page speed insights. We will be performing these activity five times to get a consistent and accurate result on page performance. The page speed insights reports on the user experience of the page on both mobile and desktop device. It measures the real user experience data and is powered by the chrome user experience report (CrUX). Pagespeed Insights reports core real user experience metrics like First Contentful Paint (FCP) , Interaction on Next Paint (INP) , Cumulative Layout Shift (CLS) and Largest Contentful Paint(LCP), it also reports for experimental metrics Time to First Byte (TTFB) as well as the deprecated metrics First Input delay (FID).

3.13.1 Core web vitals Parameters:

Google using their ongoing continuous engagement and collaboration experience with millions of developer and site owners have developed many helpful opportunities to improve user experience , web vitals is an initiative by Google to provide unified guidance for providing good experience to the website . The user experience quality have many variants , some aspects are user experience specific and some are content specific . So core web vitals will be the measuring parameter of website in this paper , Core web vitals brings all the aspect of website under one umbrella whether it's visual stability , initial loading experience, effectiveness in user interaction etc.

According the google page speed insight there are 3 metrics which can be considered and efficiently used to affect the page speed of our application those are Largest Contentful Paint (LCP) ,Cumulative Layout Shift (CLS) and Interaction on Next Paint (INP) , which replaced First Input Delay (FID) very recently .

3.13.1.1. Largest Contentful Paint (LCP)

Largest Contentful Paint is a metric which measures when a page to page navigation appears completed to a user , ideally 2.5 second for 75% of their page should load.

Figure 37: LCP Threshold

3.13.1.2. Cumulative Layout Shift (CLS)

Most of our application need to load lots of elements so browser use to load these element progressively so user can start making progress to achieve their intension without waiting for other things , but if a layout shift happens the position of already loaded , visible component will shift to other position , which will have negative impact on the overall user experience and user can commit several errors as well due to the shift . For example if user want to buy a product by clicking buy now button and due to layout shift cancel button comes on the portion of buy now button they can mistakenly click on cancel order button , this is really a worst scenario which will lead to cancellation of the order which user want to order and that will negatively affect the user experience . CLS is the metric which exactly measures the layout shift issue of our application.

Figure 38: CLS Threshold

3.13.1.3. First Input Delay (FID)

According to Kaaresoja, T.,Brewster, S., & Lantz, V. (2014) visual delay from touching button become perceived when it ranges from 70 ms to 100ms , but people rate button as slow when it gets increased from 100 ms to 150 ms . Aside from changing how the UI feels, delaying something people expect to be near-instantaneous can lead them to make errors. They may repeat an action because they think it didn't work, and the second action can have an undesirable effect. First Input Delay is a metrics which exactly measure this delay of response of the website.

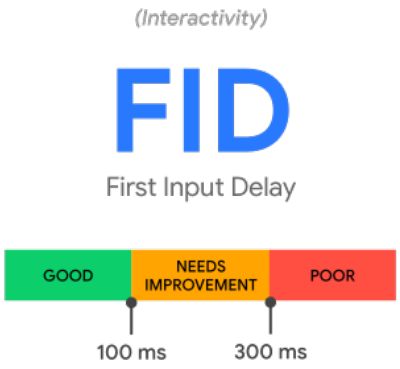


Figure 39: FID Threshold

3.13.1.4. Interaction to Next Paint (INP)

From March 2024, INP is introduced as the successor metric to First Input Delay (FID). While both are responsiveness metrics, FID only measured the input delay of the first interaction on a page. INP improves on FID by observing all interactions on a page.
It observes the latency of all click, tap , keyboard interaction within a page throughout the lifespan. Good responsiveness means a page respond quickly to interaction, when a page response to an interaction the browser present visual feedback in the next frame to show that the interaction is successful.

Figure 40: INP Threshold

	Good	Poor	Percentile
Largest Contentful Paint	<=2500ms	>4000ms	75
First Input Delay	<=100ms	>300ms	75
Cumulative Layout Shift	<=0.1	>0.25	75
Interaction on Next Paint	<=200ms	>500ms	75

Table 4: Thresholds of core web vitals which categorise performance of website

3.13.2. Pagespeed Insights Standards:

PageSpeed classifies the quality of user experience into three buckets, Good , Needs Improvement and Poor.

The distribution is divided into the categories Good , Needs Improvements and Poor , which is represented by Green , Amber and Red bars .

The metrics scores and the perf score are coloured according to these ranges:

a> 0 to 49 (red): Poor

b> 50 to 89 (orange): Needs Improvement

c> 90 to 100 (green): Good

To provide a good user experience, sites should strive to have a good score (90-100). A "perfect" score of 100 is extremely challenging to achieve and not expected. For example , taking a score from 99 to 100 needs about the same amount of metric improvement that would take a 90 to 94. The Core Web vitals metrics are INP , LCP and CLS and they may be aggregated at either the page and origin level. If aggregation of these 3 metrics is 75th of percentiles , it will be marked as Good, if aggregation has insufficient data from INP, then it

will pass the assessment if both the 75th parameters of LCP and CLS are Good. We may get variation on the page speed score , several common sources are which are responsible for these variation are local network availability , client hardware availability and content resource contention.

Metrics	Good	Needs Improvement	Poor
FCP	[0,1800 ms]	(1800 ms, 3000 ms]	Over 3000 ms
LCP	[0,2500 ms]	(2500 ms, 4000 ms]	Over 4000 ms
CLS	[0,0.1]	(0.1, 0.25]	Over 0.25
INP	[0,200 ms]	(200 ms, 500 ms]	Over 500 ms
TTFB (Experimental)	[0, 800 ms]	(800 ms, 1800 ms]	Over 1800 ms

Table 5 : Core Web Vitals Threshold

So we are having 3 pages in our application , Homepage , Aboutpage and Blogpage .We will be conducting page speed scan on 5 consecutive dates from 7th Sept 2024 - 11th Sept 2024 using page speed insights on both React and Next JS version of the application, we will be calculating the average of all the 5 page speed scan performed in each days to get the average of each core web vitals parameters, we will be collecting value of the core web vitals parameters namely : LCP,INP,CLS,TBT,FCP for both the desktop and mobile version, and we will be analysing the issues highlighted for each page in Diagnostics section .

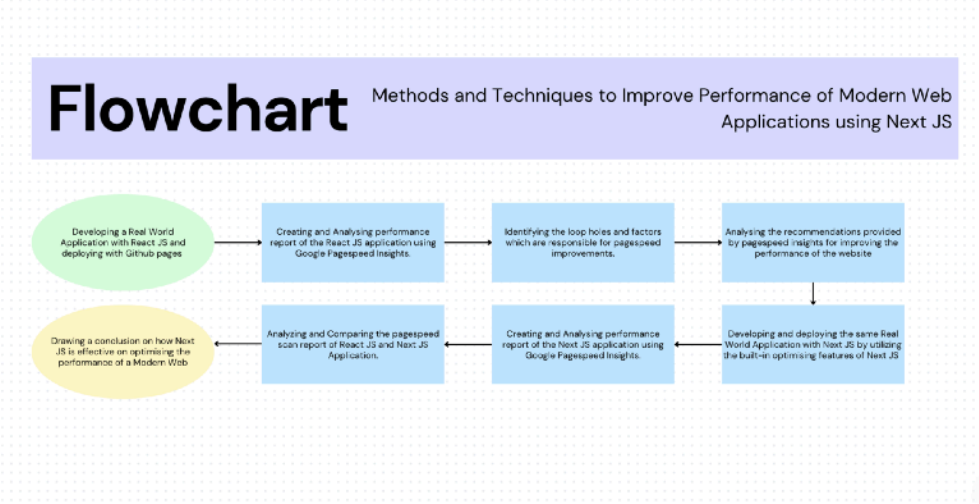


Figure 41: Flowchart of Research Methodology

3.14 Summary

In this research paper, we will be analysing how Next JS can improve the page-speed of a website. For that first we will be analysing what are the factors that affects the page-speed of a website . The main performance indicator chosen in this thesis project are the core web vitals metrics defined by Google . Web performance will be evaluated by the overall page speed score and score we are getting for each of the core web vitals metrics .Then we will explore and analyse how Next JS can leverage the page-speed of a website. After that we will be conducting page-speed scan on a reference static site using Google Page-speed insights/LightHouse , which is a tool for improving the quality of web pages and is able to run a variety of test against a webpage while monitoring various performance like the core web vitals and speed index.Test will be conducted 5 times on Google Chrome browser using MacBook Air M1 2020. The important metrics we will consider while comparing and measuring the website performance are :

- First Contentful Paint (FCP): measures the time required from starting page loading to rendering any part of the page .
- Largest Contentful Paint (LCP): measures the time required to start loading the page to rendering the largest text block on the screen.
- Interaction to Next Paint (INP): measures the latency of every action in the site like tap, keyboard interaction , click on the page . It measures the response time of the every action in the website.
- Total Blocking Time (TBT): measures how long the main thread is blocked which prevents user interaction and website responsiveness .
- Cumulative Layout Shift (CLS): measures the unexpected layout shift or sudden shifting of the components in the site .
- Time to First Byte (TTFB): measures the time taken for the network to respond to the first byte of the resource .

Then after analysing the scan report we will be developing the same application using Next JS and will conduct another page-speed scan on the application which is developed using Next JS . Finally we will compare and evaluate both the page speed scan report to draw a conclusion on how Next JS is impacting the page speed of the application .

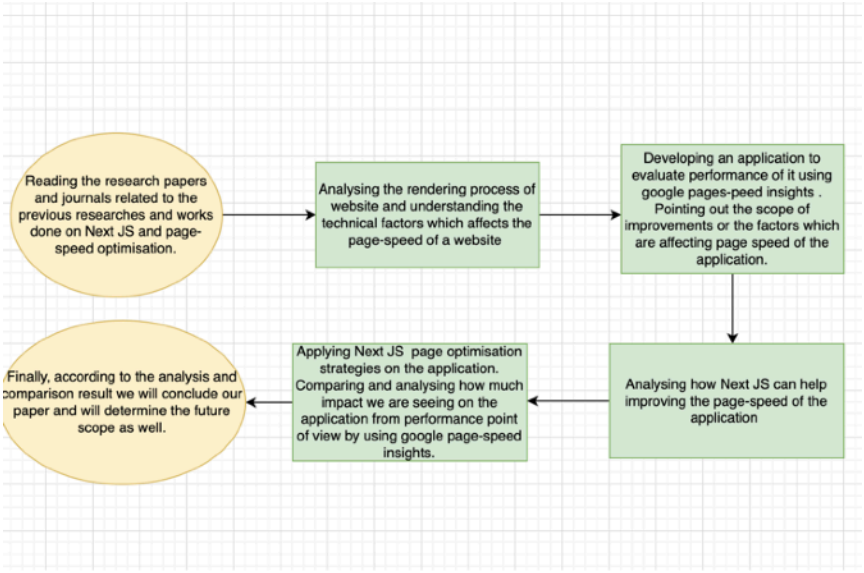


Figure 42: Work flow of the research plan

3.15 Required tools

We will be using Google Page speed Insights for generating the website page speed report which we will be analysing.We will be using Node Js as the JavaScript runtime environment ,Visual Studio code as the code editor , Git as the distributed version control system , Github as software development platform for storing , tracking and collaboration of software development, Github pages as static site hosting service to deploy application, Mendeley for managing citation of our research paper . We will be using MacBook Air M1 2020 having Apple M1 chip with Mac OS , MacOS Sonoma 14.0 version for implementing all the tasks for this research paper.

CHAPTER-4

ANALYSIS

4.1 Introduction :

We will be running page speed scan for both the React and Next version of the application with page speed insights for desktop and mobile view , the parameters or metrics which we will be measuring to analyse the page speed of the website are Largest Contentful Paint , First Contentful Paint , Cumulative Layout Shift & Total Blocking Time. We will also go through the diagnostic section of the page speed report to determine what are the assets and factors which are affecting the page speed of the website , and how we can remediate or resolve the issues highlighted for the website . We will run page speed scan from 7th Sept to 11th Sept 2024. We will be using MacBook Air M1 2020 having Apple M1 chip with Mac OS , MacOS Sonoma 14.0 version and Google chrome browser in incognito mode to run the page speed scan. Our Application is having 3 pages those are Homepage , Blogpage and Aboutpage. We will be analysing each of these page and will discuss on the xrecommendation suggested in the diagnostic section for each page.

4.2 React Application Analysis:

We will be analysing all the speed page metrics score for the three pages of the react application both for mobile and desktop view.

4.2.1 First Contentful Paint:

First Contentful Paint measures the time required from starting page loading to rendering any part of the page. For mobile view the average FCP score for Homepage is 920 millisecond . For desktop the average FCP score for Homepage is 260 millisecond.
For mobile view the average FCP score for Aboutpage is 800 millisecond . For desktop view of the application the average FCP score for Aboutpage is 200 millisecond. For mobile view the average FCP score for Blogpage is 800 millisecond . For desktop view of the average FCP score for Blogpage is 200 millisecond .

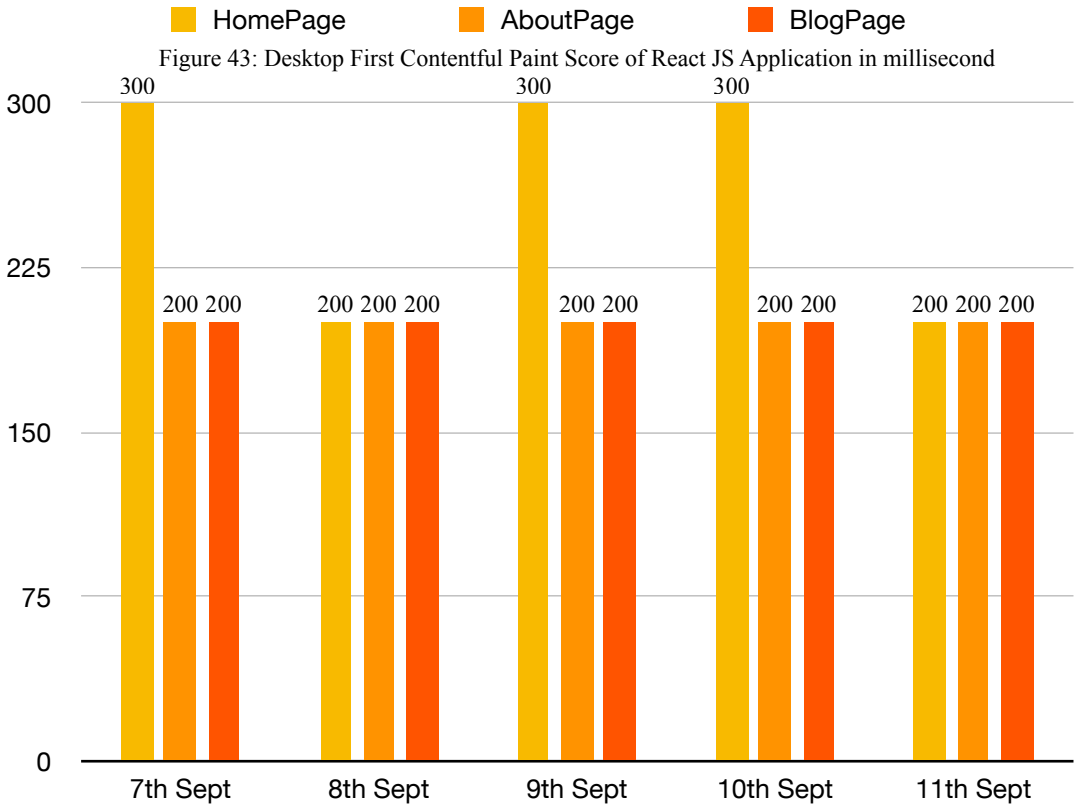


Figure 43: Desktop First Contentful Paint Score of React JS Application in millisecond

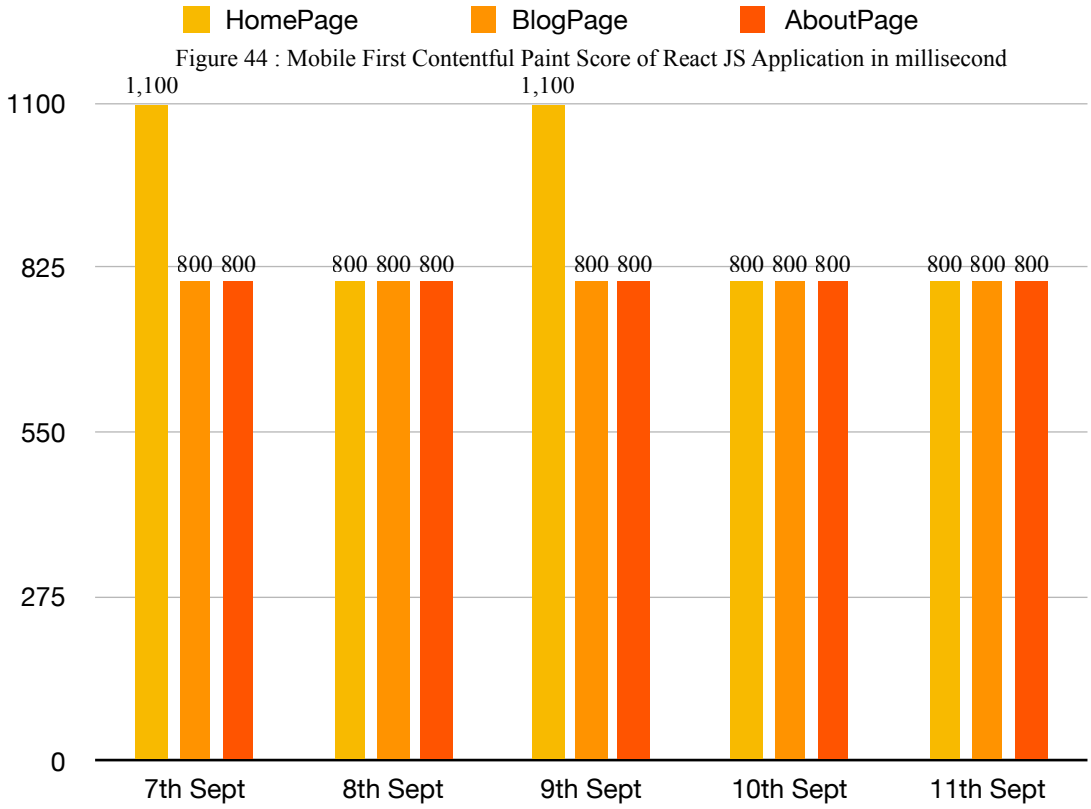


Figure 44 : Mobile First Contentful Paint Score of React JS Application in millisecond

4.2.2 Largest Contentful Paint:

Largest contentful paint measures the time required to start loading the page to rendering the largest text block on the screen. For mobile view of the application , the average LCP score for Homepage is 1s 440ms millisecond.For desktop the average LCP score for the Homepage is 380 millisecond .For mobile view of the application, the average LCP score for Aboutpage is 1200 millisecond, and for desktop view of the application the average LCP score for Aboutpage is 300 millisecond. For mobile view of the application , the average LCP score for Blogpage is 1340 millisecond . For desktop view of the application , the average LCP score for Blogpage is 520 millisecond .

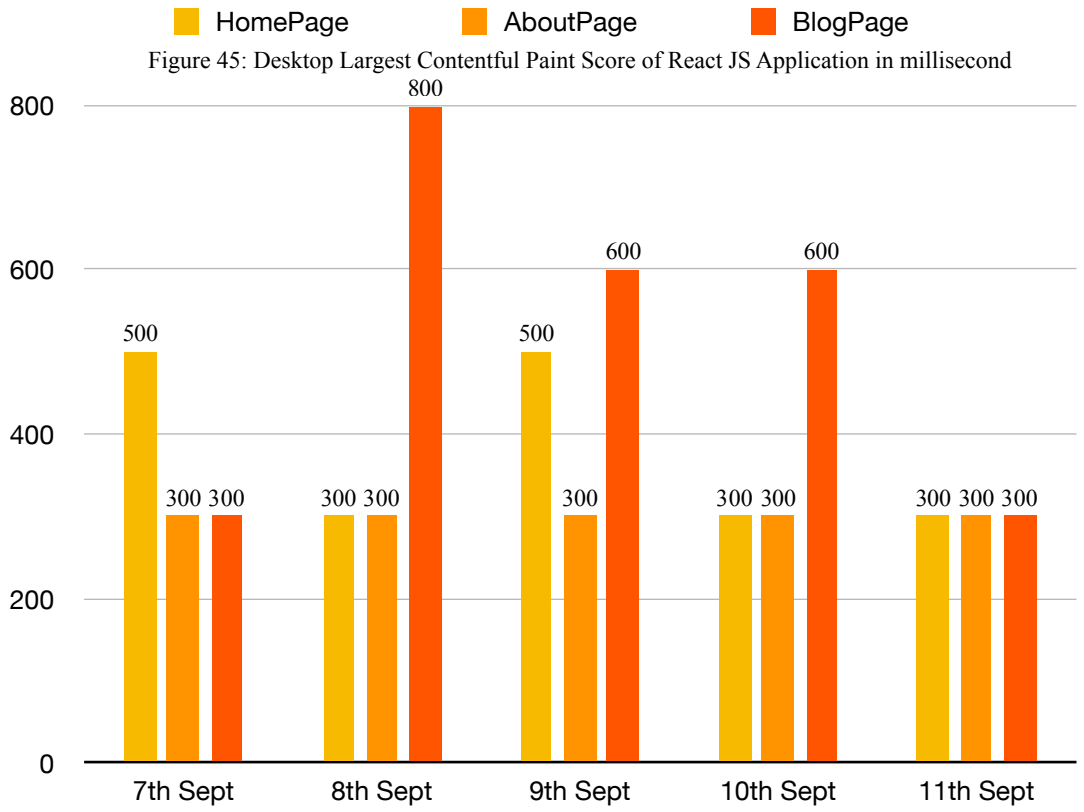


Figure 45: Desktop Largest Contentful Paint Score of React JS Application in millisecond

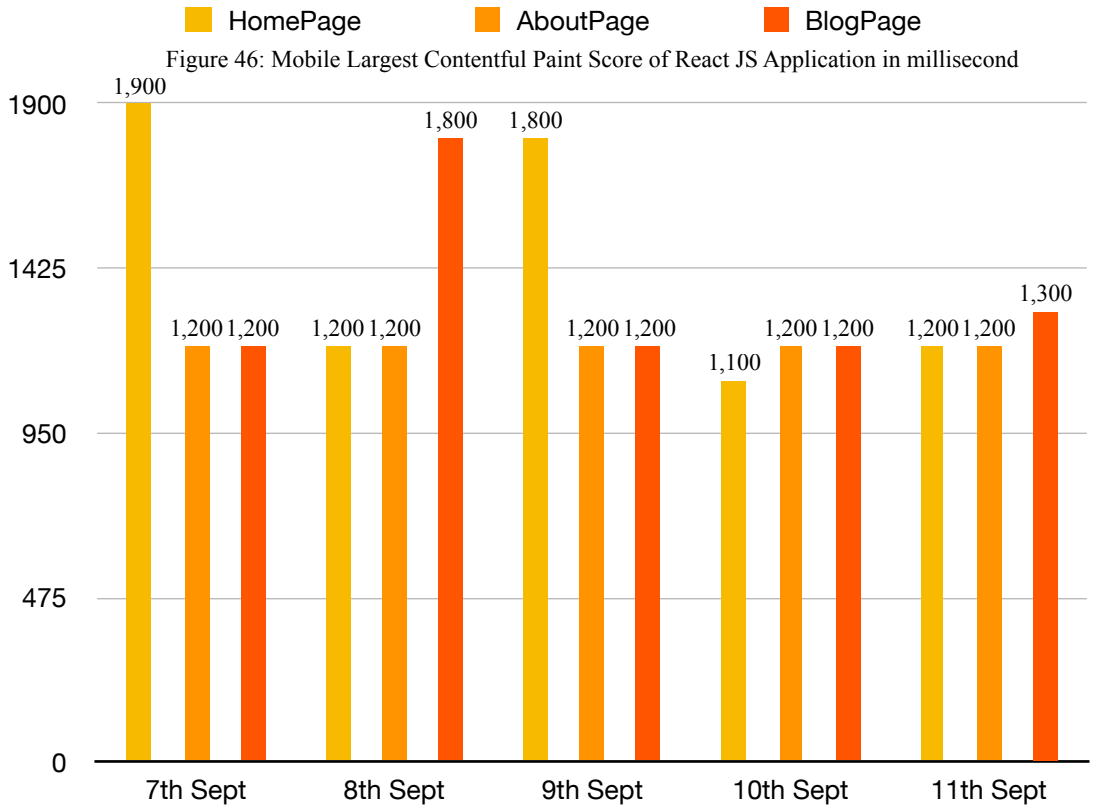


Figure 46: Mobile Largest Contentful Paint Score of React JS Application in millisecond

4.2.3 Total Blocking time :

Total Blocking time measures how long the main thread is blocked which prevents user interaction and website responsiveness . For mobile view of the application , the average TBT score for Homepage is 34 millisecond, for desktop view of the application the average TBT score for Homepage is 0 millisecond. For mobile view of the application , the average TBT score for Aboutpage is 48 millisecond , whereas for desktop the average TBT score for Aboutpage is 0 millisecond. For mobile view of the application , the average TBT score for Blogpage is 68 millisecond and for desktop the average TBT score for Blogpage is 0 millisecond .

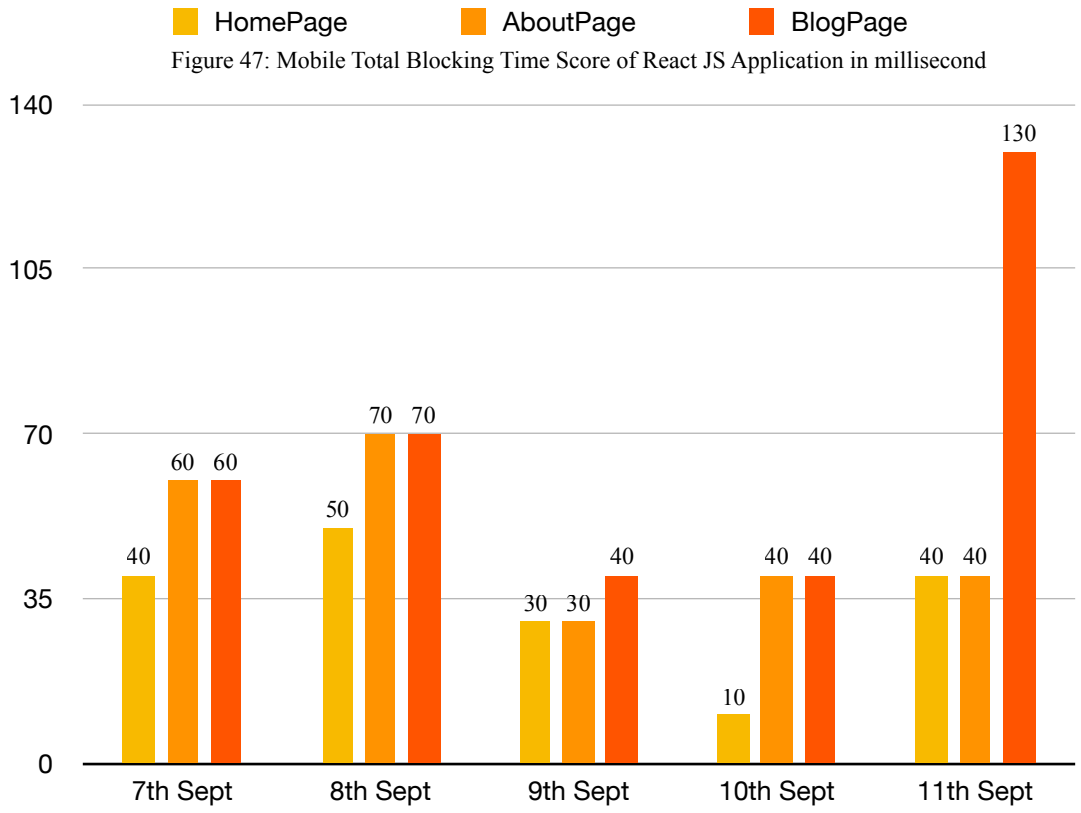


Figure 47: Mobile Total Blocking Time Score of React JS Application in millisecond

4.2.4 Overall Summary :

In diagnostics section , it is recommended to eliminate render blocking resource , these resources are blocking the first paint of our page, and it is suggesting to deliver critical JS/ CSS inline and defer the loading of all non critical JS/styles, it's highlighting the main.css used in our application as render blocking resource and are suggested to optimise to improve the Largest Contentful Paint and First Contentful Paint score of the website . Nothing major is highlighted for the factors affecting the Total blocking time , but it is something which is highly influenced by FCP and LCP score and vice versa . CLS measures the unexpected layout shift or sudden shifting of the components in the site . From CLS point of view we are having good score since the application is not having multiple folds and it consists only one fold, but in diagnostic section it is recommended to set an explicit width and height on image elements to reduce the layout shift and improve CLS and it's recommended to use next image instead of normal image tag .It is also recommended to serve image in next-gen formats to improve LCP and FCP . Image formats like Webp and Avif often provides better compression than Png and Jpeg, which means faster download and less data consumption. It also suggested to use properly sized images to improve LCP and FCP , serve image that are appropriately sized saves cellular data and improve load time . To improve FCP and LCP it is suggested to use efficiently encode images that loads faster and consumes less cellular data .

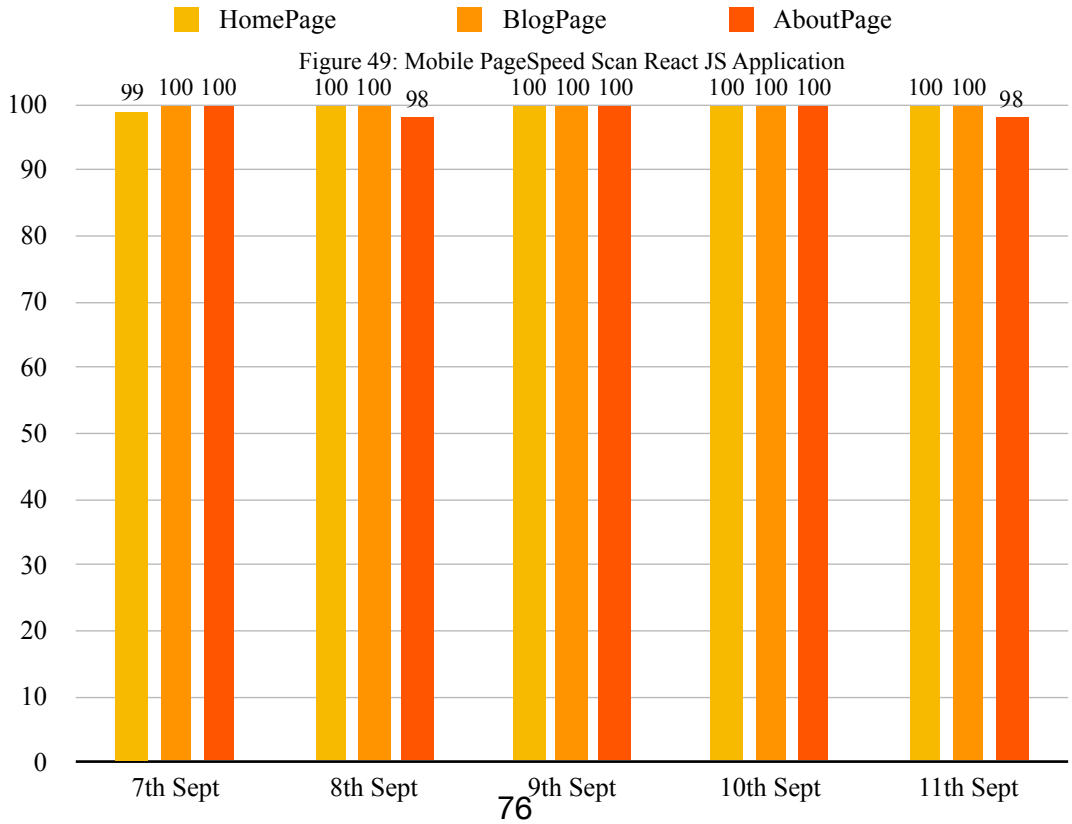


Figure 49: Mobile PageSpeed Scan React JS Application

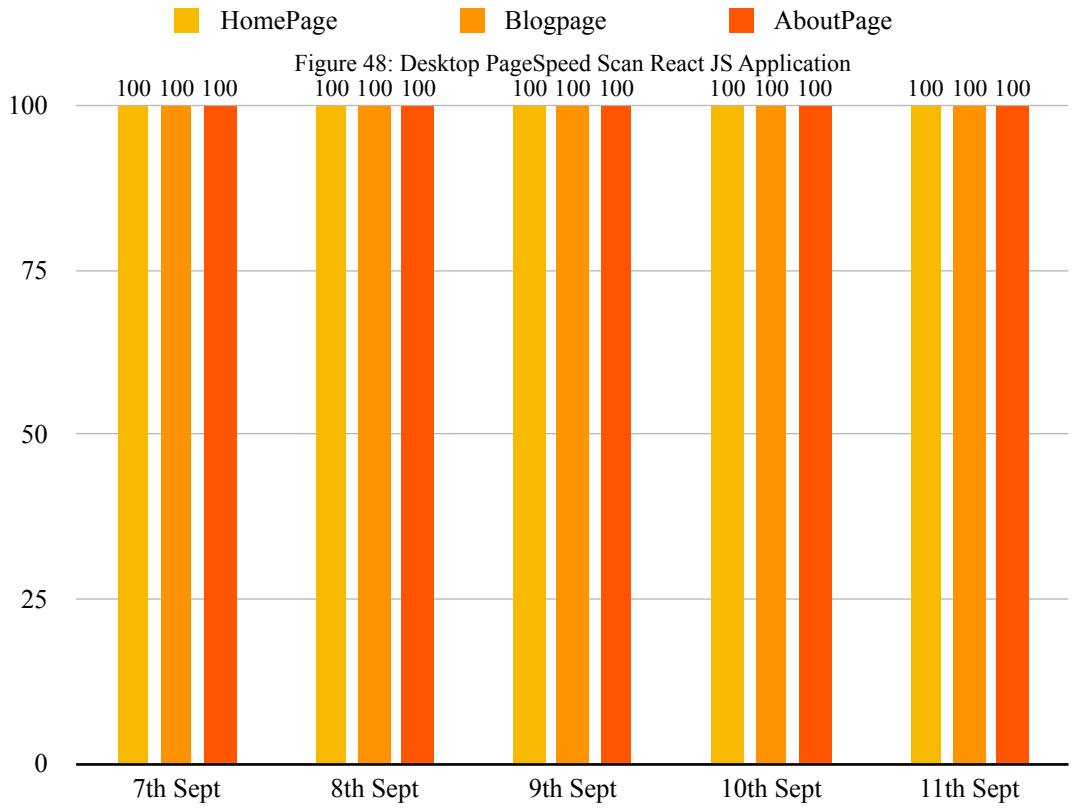


Figure 48: Desktop PageSpeed Scan React JS Application

4.3 Next Application Analysis:

We will be analysing all the speed page metrics score for the three pages of the next application both for mobile and desktop view.

4.3.1 First Contentful Paint:

First Contentful Paint measures the time required from starting page loading to rendering any part of the page. For mobile view the average FCP score for Homepage is 800 millisecond . For desktop the average FCP score for Homepage is 200 millisecond.
For mobile view the average FCP score for Aboutpage is 800 millisecond . For desktop view of the application the average FCP score for Aboutpage is 200 millisecond. For mobile view the average FCP score for Blogpage is 800 millisecond . For desktop view of the average FCP score for Blogpage is 200 millisecond .

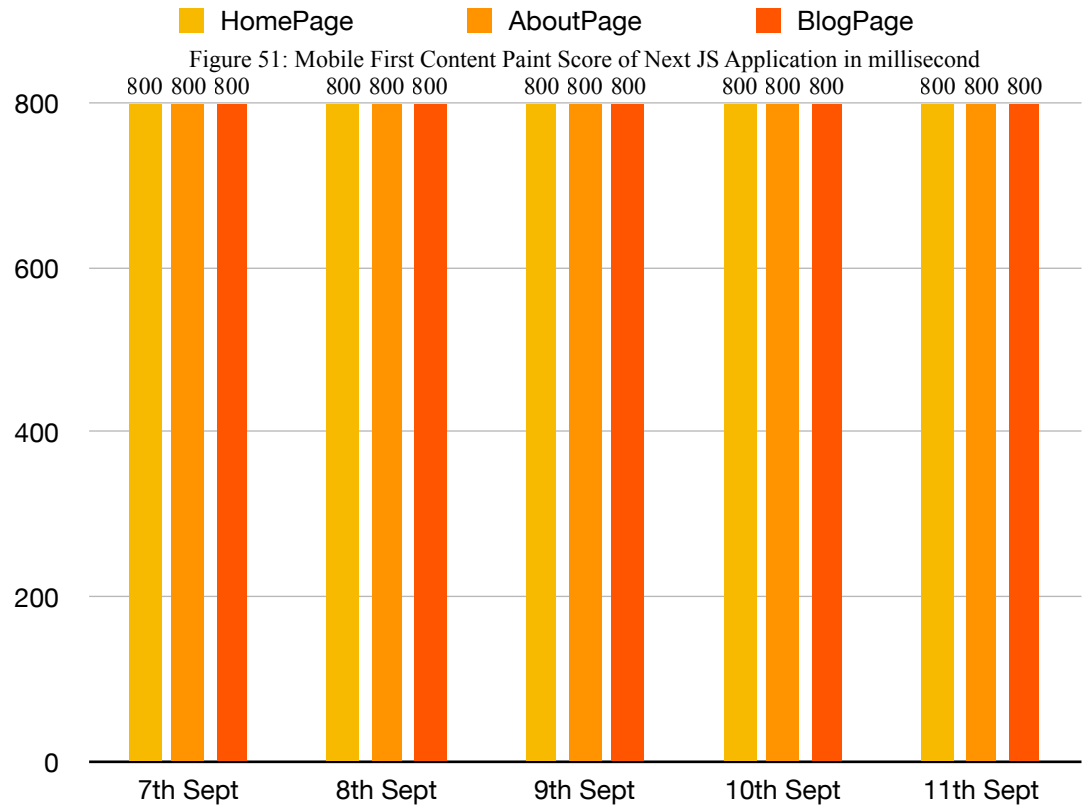


Figure 51: Mobile First Content Paint Score of Next JS Application in millisecond

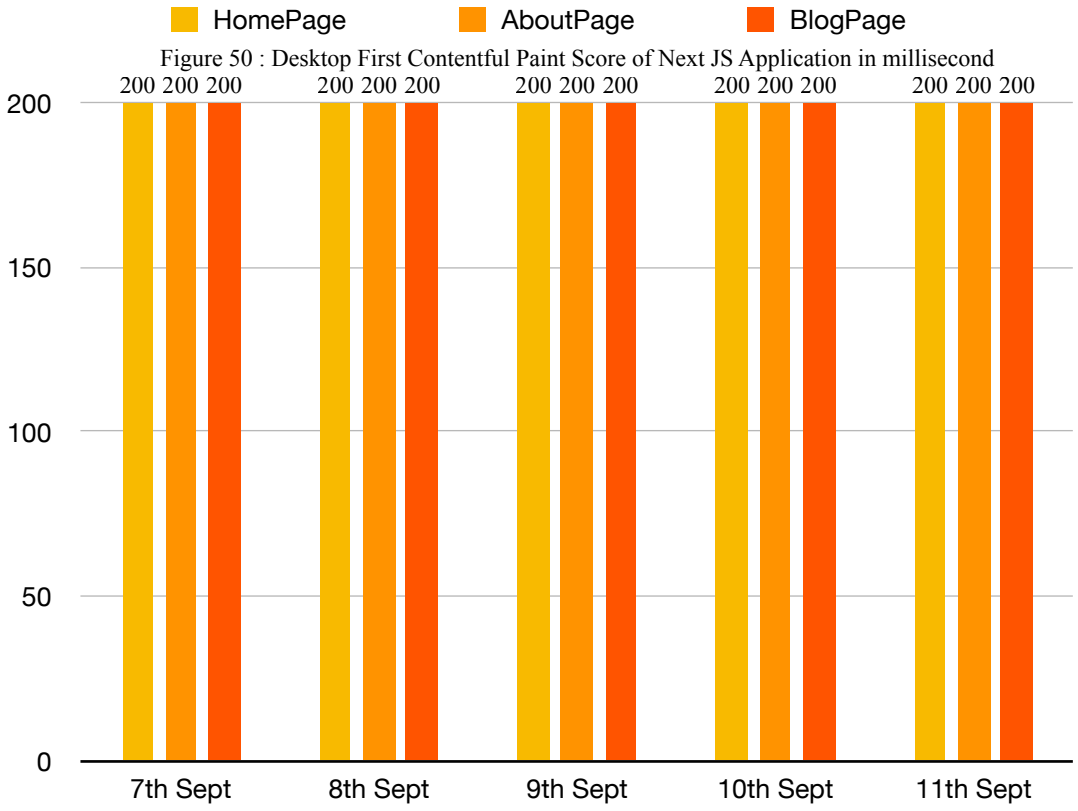


Figure 50 : Desktop First Contentful Paint Score of Next JS Application in millisecond

4.3.2 Largest Contentful Paint:

Largest contentful paint measures the time required to start loading the page to rendering the largest text block on the screen. For mobile view of the application , the average LCP score for Homepage is 1140ms millisecond.For desktop the average LCP score for the Homepage is 280 millisecond .For mobile view of the application, the average LCP score for Aboutpage is 980 millisecond, and for desktop view of the application the average LCP score for Aboutpage is 300 millisecond. For mobile view of the application , the average LCP score for Blogpage is 1320 millisecond . For desktop view of the application , the average LCP score for Blogpage is 420 millisecond .

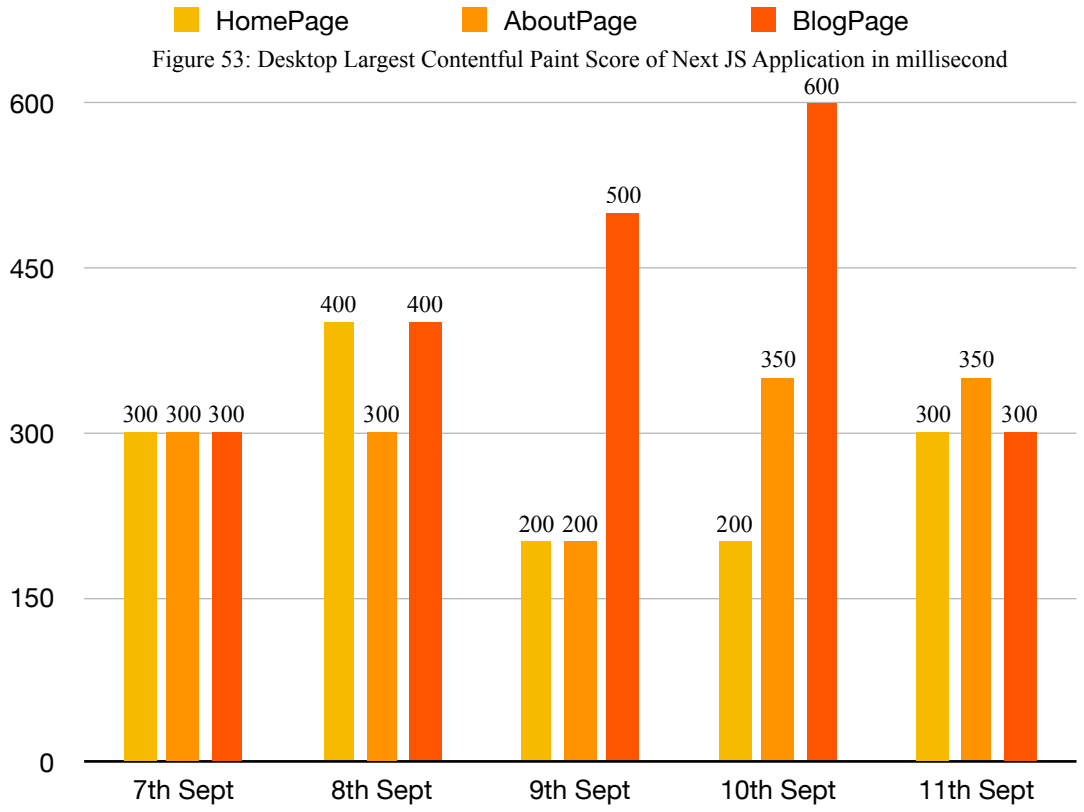


Figure 53: Desktop Largest Contentful Paint Score of Next JS Application in millisecond

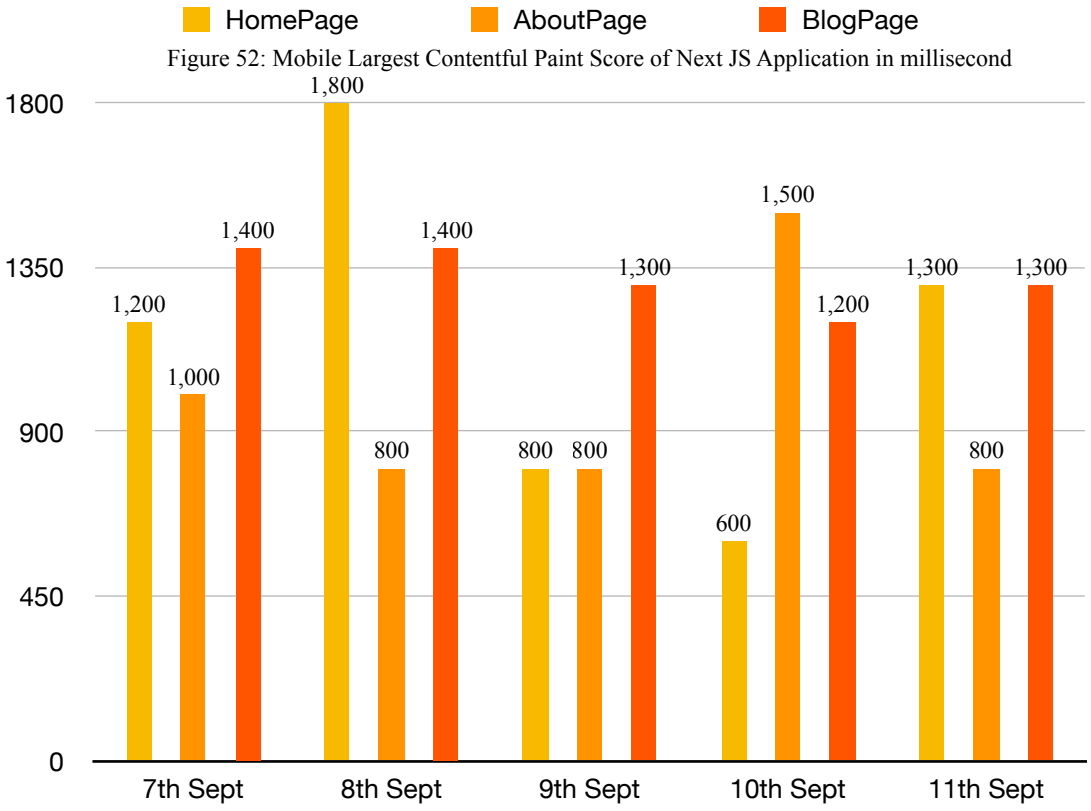


Figure 52: Mobile Largest Contentful Paint Score of Next JS Application in millisecond

4.3.3 Total Blocking time :

Total Blocking time measures how long the main thread is blocked which prevents user interaction and website responsiveness . For mobile view of the application , the average TBT score for Homepage is 62 millisecond, for desktop view of the application the average TBT score for Homepage is 0 millisecond. For mobile view of the application , the average TBT score for Aboutpage is 64 millisecond , whereas for desktop the average TBT score for Aboutpage is 56 millisecond. For desktop view of the application , the average TBT score for Blogpage is 72 millisecond and for mobile the average TBT score for Blogpage is 104 millisecond .

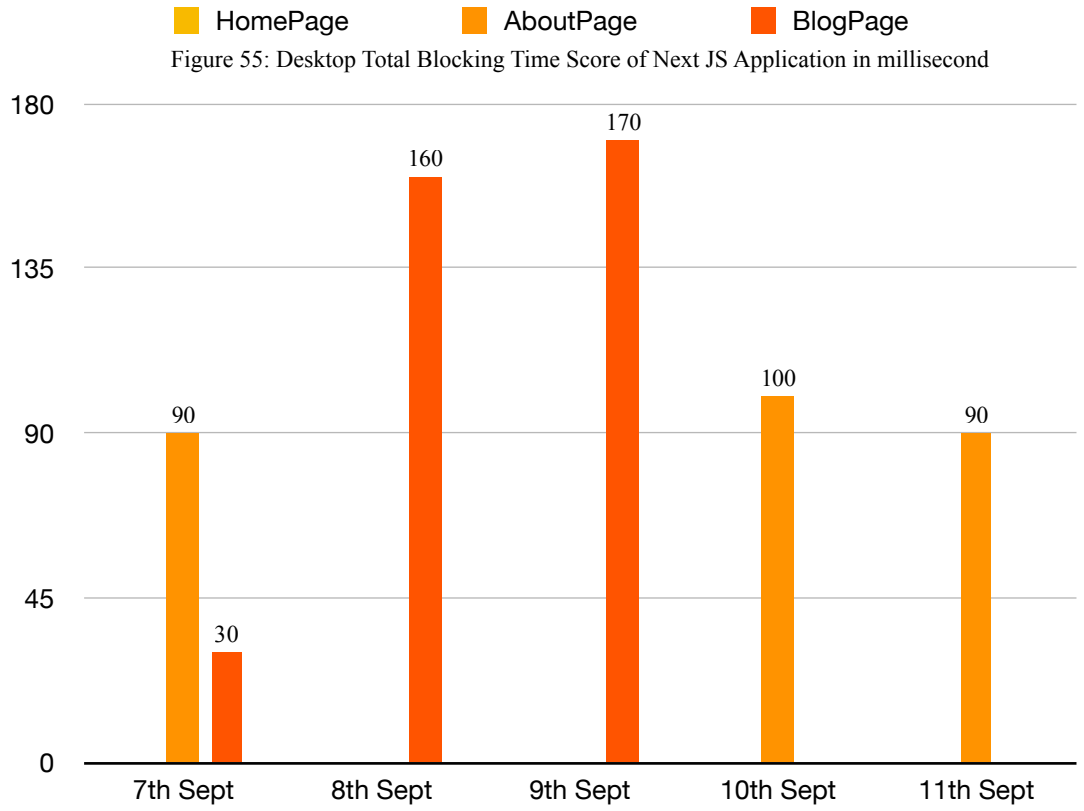


Figure 55: Desktop Total Blocking Time Score of Next JS Application in millisecond

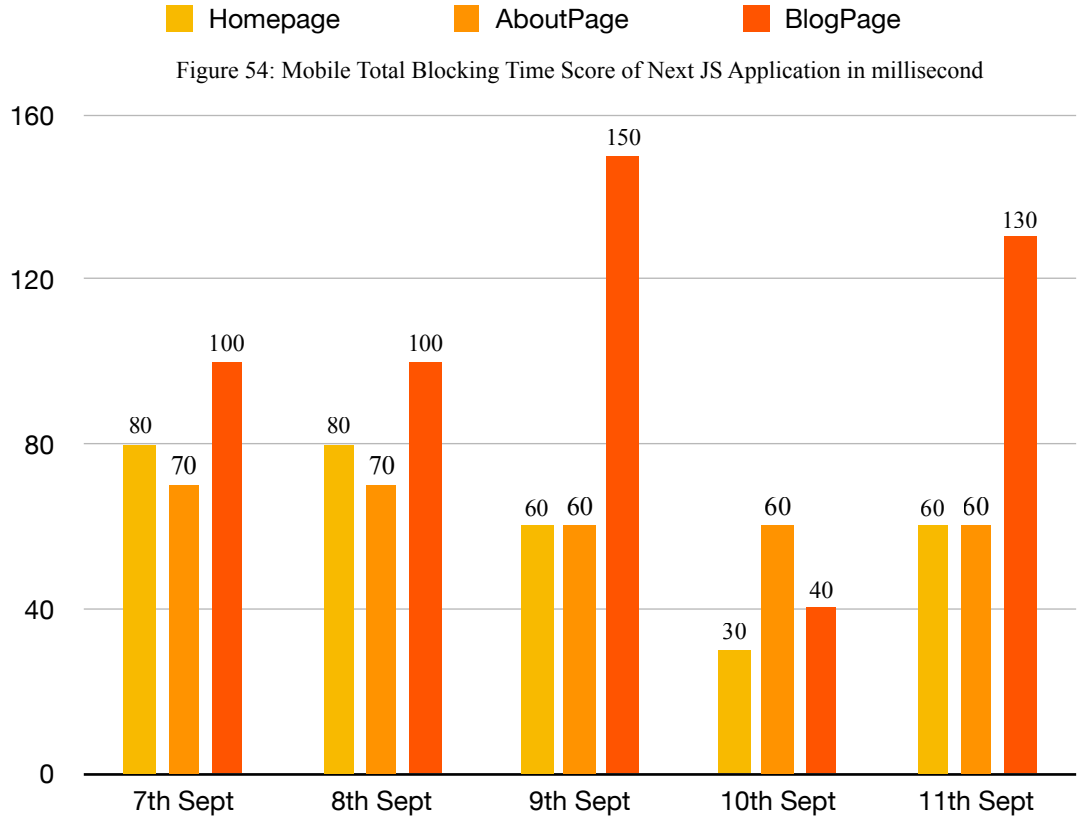


Figure 54: Mobile Total Blocking Time Score of Next JS Application in millisecond

4.3.4 Overall Summary :

In diagnostics section , it is recommended to server static assets with a efficient cache policy to improve overall speed of the application. A long cache lifetime can speed up repeat visits to our pages and suggested to configure caching for immutable assets and server side rendering (SSR) pages. To improve the LCP and FCP of the application is to recommended to avoid serving legacy JS to modern browsers. Polyfills and transforms enable browsers to use Next JS features, however many JS are not necessary for modern browsers . For our bundle javascript , adopt a modern script deployment strategy using module/nomodule feature detection to reduce the amount of code shipped to modern browsers, while retaining support for legacy browsers. Reducing the unused JS is very helpful to improve the LCP and FCP of the website. It suggested to reduce unused JS and defer loading scripts until they are required to decrease bytes consumed network activity. To reduce the load on main thread it is suggested to avoid numerous network payload , large network payloads and chunks costs real money and are highly correlated with long load times. CLS measures the unexpected layout shift or sudden shifting of the components in the site . From CLS point of view we are having good score since the application is not having multiple folds and it consists only one fold.

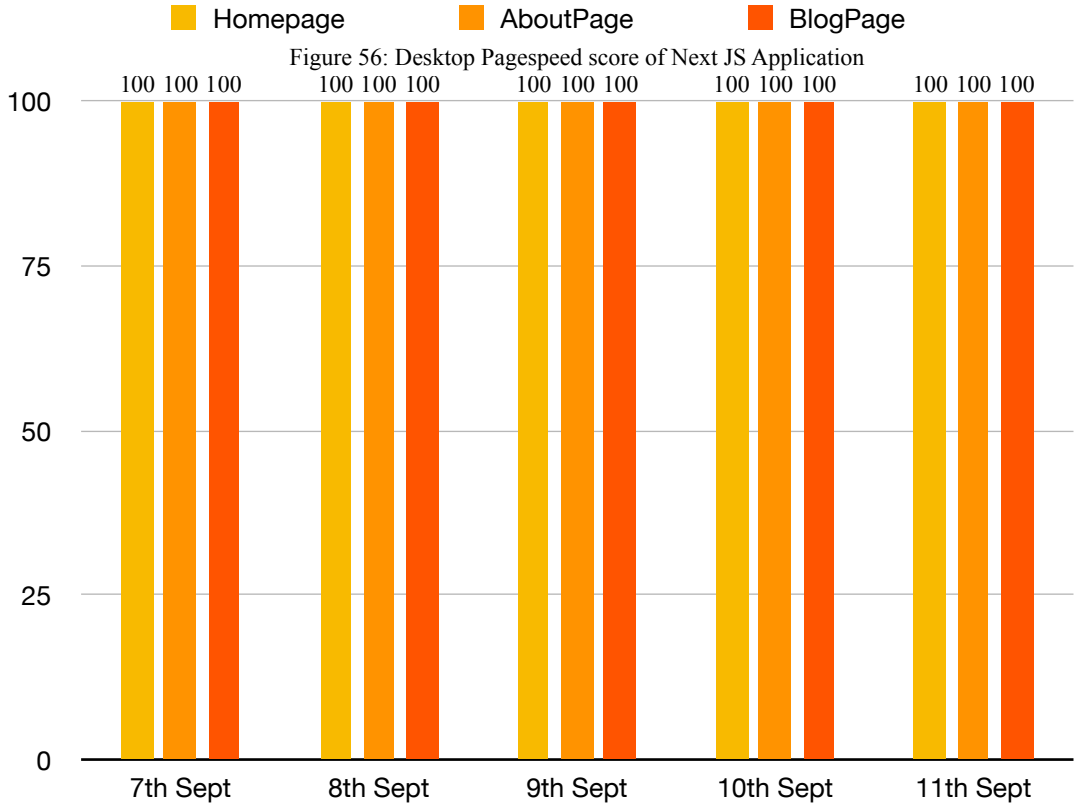


Figure 56: Desktop Pagespeed score of Next JS Application

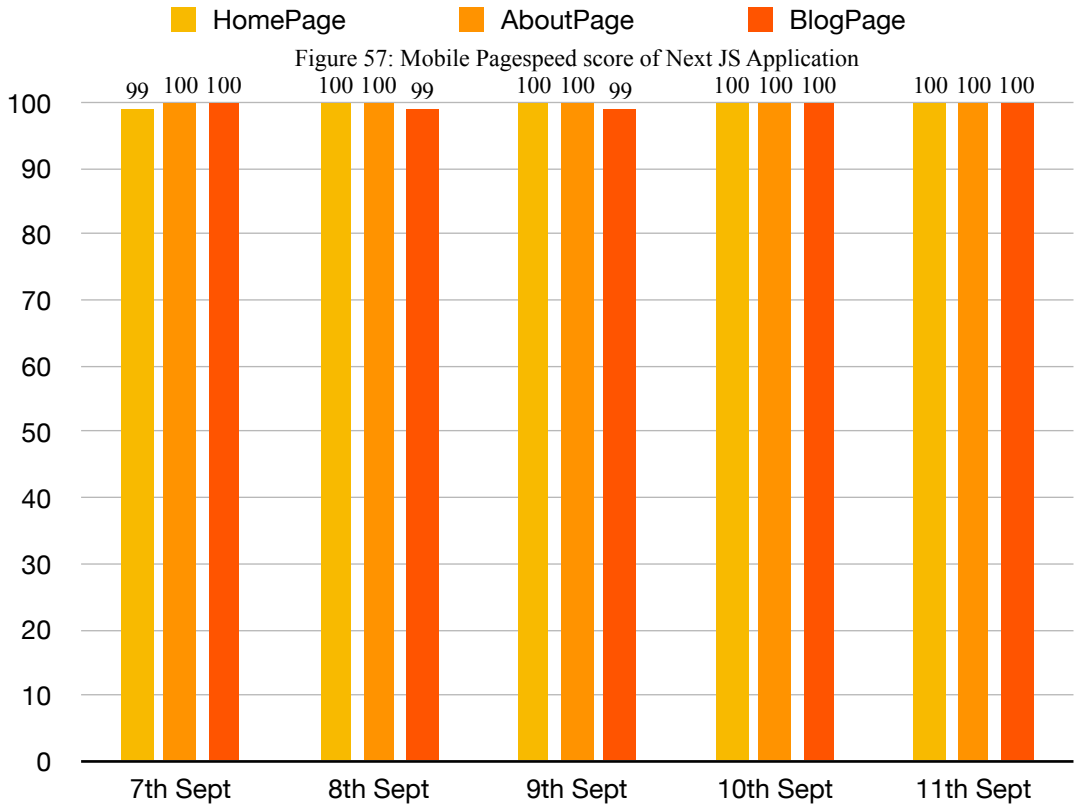


Figure 57: Mobile Pagespeed score of Next JS Application

CHAPTER-5

RESULTS & DISCUSSIONS

5.1 Introduction:

The main aim of this research is to evaluate the effectiveness of Next js for improving performance of the website. We will be analysing each of these page and will discuss on the recommendation suggested in the diagnostic section for each page to meet the objective of our research paper.

5.2 Factors affecting the pagespeed of the application:

We have analysed the page speed report generated using pagespeed insights for both react application and next application and have identified the factors which are responsible for slowing down the page speed of our application highlighted in diagnostic section of the report .

5.2.1 Eliminate Render Blocking Resources:

The page speed scan report has clearly highlighted the factors which influence the website performance . So when we scanned the react application using pagespeed insights. For react application it has highlighted main.css as the render blocking resource. Rendering blocking resource are the resource which prevents the content from rendering , which can result long waiting time for the visitors. If we can optimise our render blocking resource of our application we can improve the LCP and FCP of our website .

5.2.2 Set an explicit width & height on image element:

The pagespeed scan report has clearly highlighted that images used in the react application are not having explicit width and height. If image are declared with height and width attribute or usually resized using css , when this happens browser can only determine their dimension and allocated space for them . When it starts downloading the unsized images which can cause unexpected layout shift. If we can optimise our render blocking resource of our application we can improve the LCP , FCP and CLS of our website.

5.2.3 Serve image in next-gen formats:

The pagespeed scan report recommended to use Next-gen formats of the images in our react application . Next-gen format such as Webp or Avif provide superior compression features compared to all other web image formats. Avif image format is an open-source image format for storing image which supports lossy and lossless compression to produce high optimised image which can make 50% save on total file size . Webp image also provides superior lossless and lossy compression for images on the website which could be 25% to 34% smaller than a comparable JPEG file. By serving image in next-gen formats we can improve FCP and LCP of our website.

5.2.4 Properly Size image:

The page speed scan report tells us that some of the page images of our react application are served with significantly larger dimension than needed, which can have some adverse effect on the user's experience as images takes longer time to load as it is heavy , browser needs additional time to resize the image , the resulting visual quality might also suffer. Our website should never serve images that are larger than the version that's rendered on the user screen. If the rendered size is at least 4 KiB smaller than the actual size then image fails the audit. By using properly sized images we can improve the FCP and LCP score of our website.

5.2.5 Efficiently encode images:

The page speed report lists all the unoptimised images , it collects all the JPG and BMP images on the page, sets each image compression level to 85, and then compare the original version with the compressed version , if the potential savings are 4KiB or greater , it highlight as optimisable . We can optimise image by using webpages and avif format of the images, lazy loading images , using responsive images . For these remediation can be implemented by using Next image , it can handle these optimisation which are out of box for us , it provides optimisation, lazy loading images , avoid CLS etc. By using efficient encoded images we can improve the FCP and LCP score of our website.

5.2.6 Defer Offscreen images:

The page speed report recommended to defer images which are offscreen or the images which are not on the first fold of the website .This can be achieved by lazy loading non critical assets and images and dynamically importing the components used in the website. Lazy loading is a web development technique that loads images when they are about to visible on the page. Viewport is the loaded area that is in sight of the viewer at one time . Whether your visitor browse via laptop on mobile device, only a certain part of the page is visible at one time to begin with. If we defer or pause the download of all offscreen images on the pages simultaneously, the images are download only when the user scroll down and goes to other folds of the webpage. Deferring the offset images helps us reducing the burden on the server and allow them to display pages fast to visitors, by deferring images we can reduce load time , bandwidth conservation also comes down significantly which results better SEO scores and visitor experience. By deferring the offscreen images we can improve the LCP and FCP score of our website .

5.3 Impact of Next JS on the pagespeed of the application:

Now we will be comparing the page speed report of our React and Next application to identify the impact of Next JS on the page speed of the application.

5.3.1 Impact of Next JS on First Contentful Paint:

The FCP measures the time from when the user first navigated to the page to when any part of the page content is rendered on the screen. To improve FCP for a specific site we have to reduce and eliminate the render blocking resources , we have to set an explicit width and height on images used in our application , we have to serve images in next-gen format , and we have to use properly size images. Next JS comes with built-in image component called next/image which can handle the optimisations which are out of box for us. So in our next application we have used next/image instead of normal image tag , where we preloaded the images which are rendering on the first fold setting priority as true, next/image also optimised the images , it have both built-in and custom method to optimise the image. Using next/image we can defer the offscreen images by lazy loading them. We are also importing the components dynamically using next/dynamic to avoid unnecessary downloading and rendering of the offscreen assets. We are using next/link instead of anchor tag, next/link by default prefetches the pages whose links are in the viewport. So by implementing built-in Next JS optimisation features in our Next JS application we can see improvement in the First Contentful Paint score of our application.

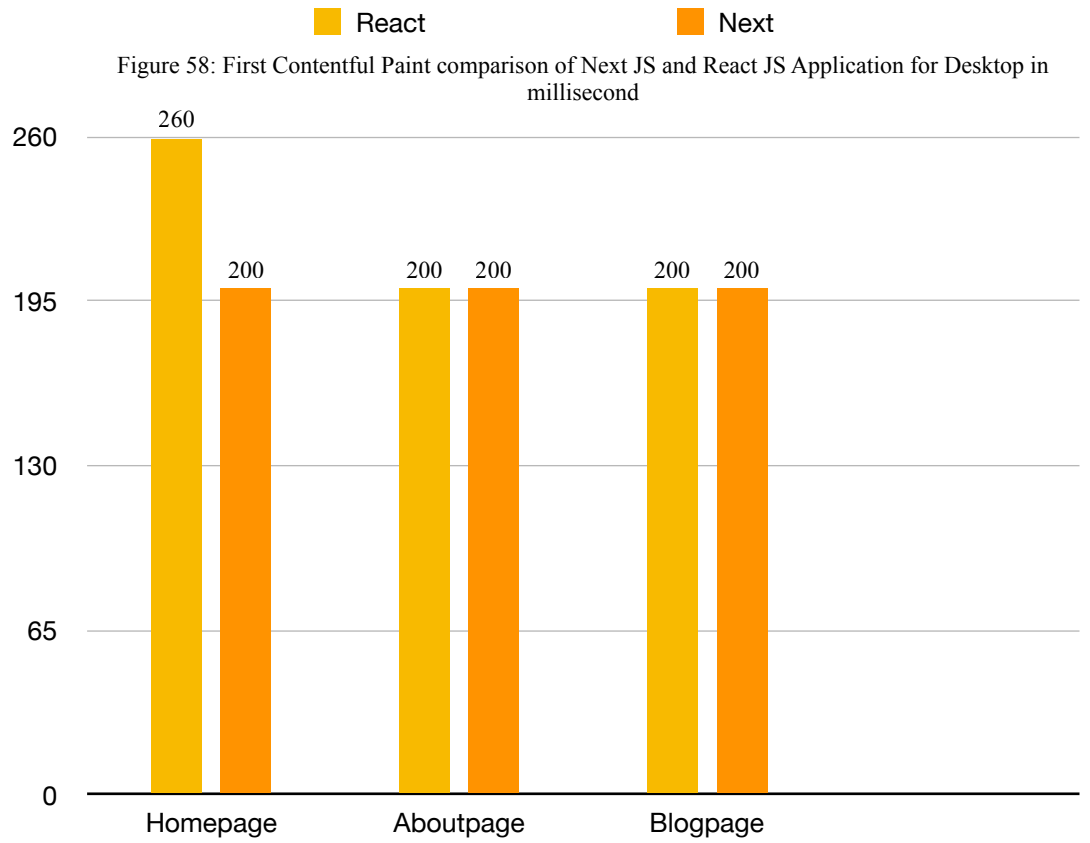


Figure 58: First Contentful Paint comparison of Next JS and React JS Application for Desktop in millisecond

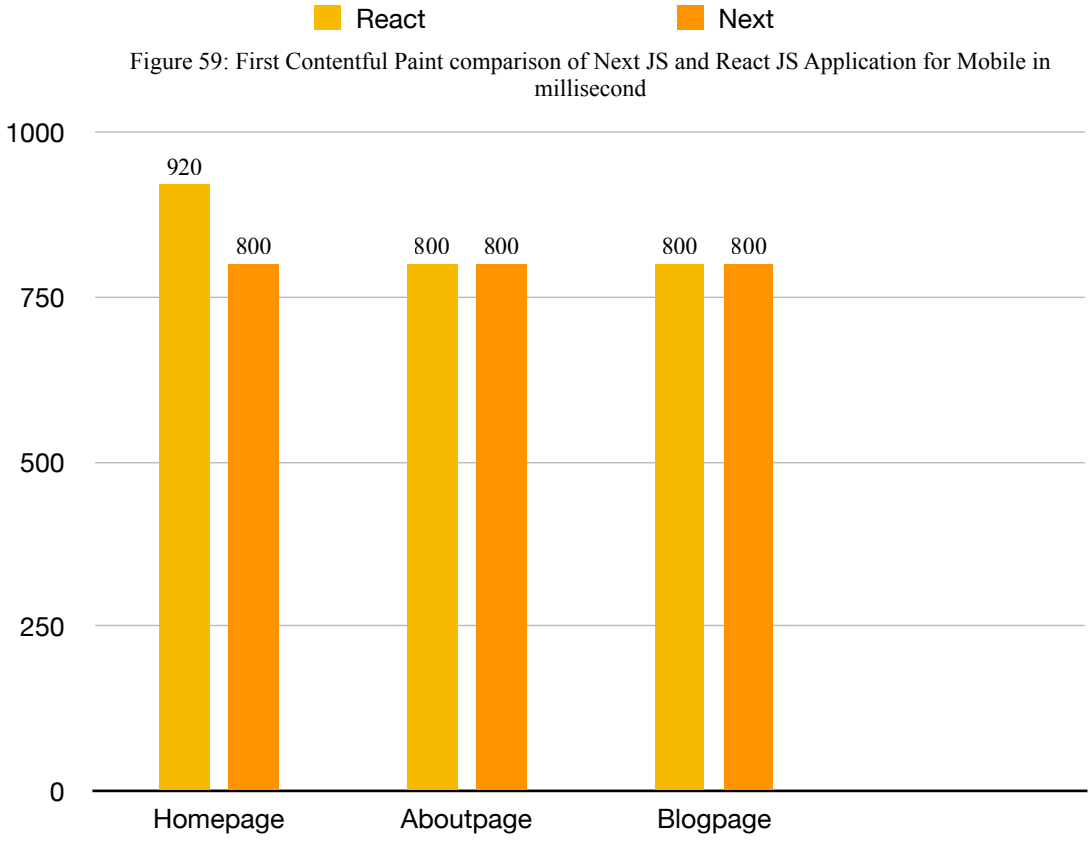


Figure 59: First Contentful Paint comparison of Next JS and React JS Application for Mobile in millisecond

5.3.2 Impact of Next JS on Largest Contentful Paint:

LCP reports the render time of the largest image, text block, or video visible in the viewport, relative to when the user first navigated to the page.To improve LCP for a specific site we have to reduce and eliminate the render blocking resources , we have to optimize the images , minify code , preload critical resources , defer parsing of Javascript , use properly sized images , implement lazy load etc . In the next application we are using Next-gen format images. Next-gen format such as Webp or Avif provide superior compression features compared to all other web image formats. Avif image format is an open-source image format for storing image which supports lossy and lossless compression to produce high optimised image which can make 50% save on total file size . Webp image also provides superior lossless and lossy compression for images on the website which could be 25% to 34% smaller than a comparable JPEG file. Using next/script we can optimise , preload or defer any third party script. Using next/fonts we can optimise the fonts used in our application . Next JS also help us to code splitting since , it leverages dynamic imports using import( ) to split code into smaller bundles. Instead of loading all JavaScript files upfront, dynamic imports allow components or modules to be loaded asynchronously when they are needed. In Next JS, code splitting is done automatically based on routes. Each page component in the pages directory becomes a separate chunk of JavaScript. When a user navigates to a specific route, only the JavaScript necessary for that route is loaded. By leveraging those Next Js optimisation features we can see improvement in Largest Contentful Paint score of the application.

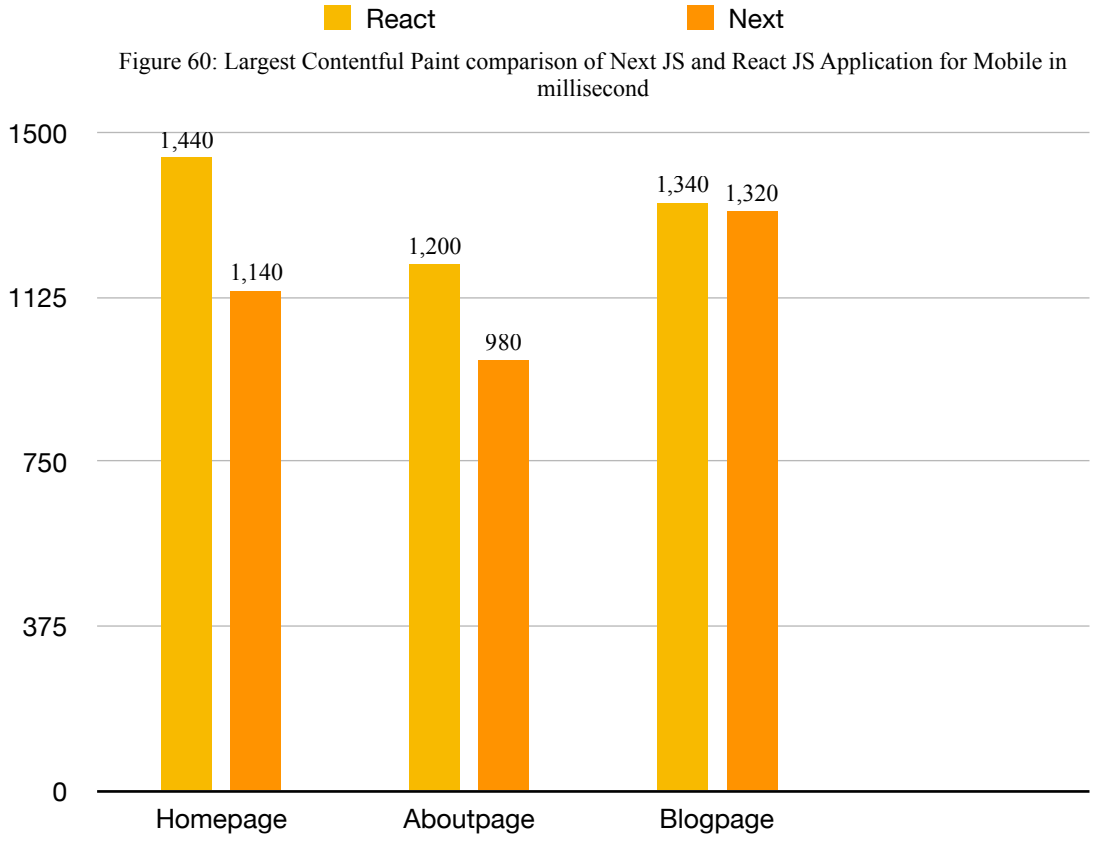


Figure 60: Largest Contentful Paint comparison of Next JS and React JS Application for Mobile in millisecond

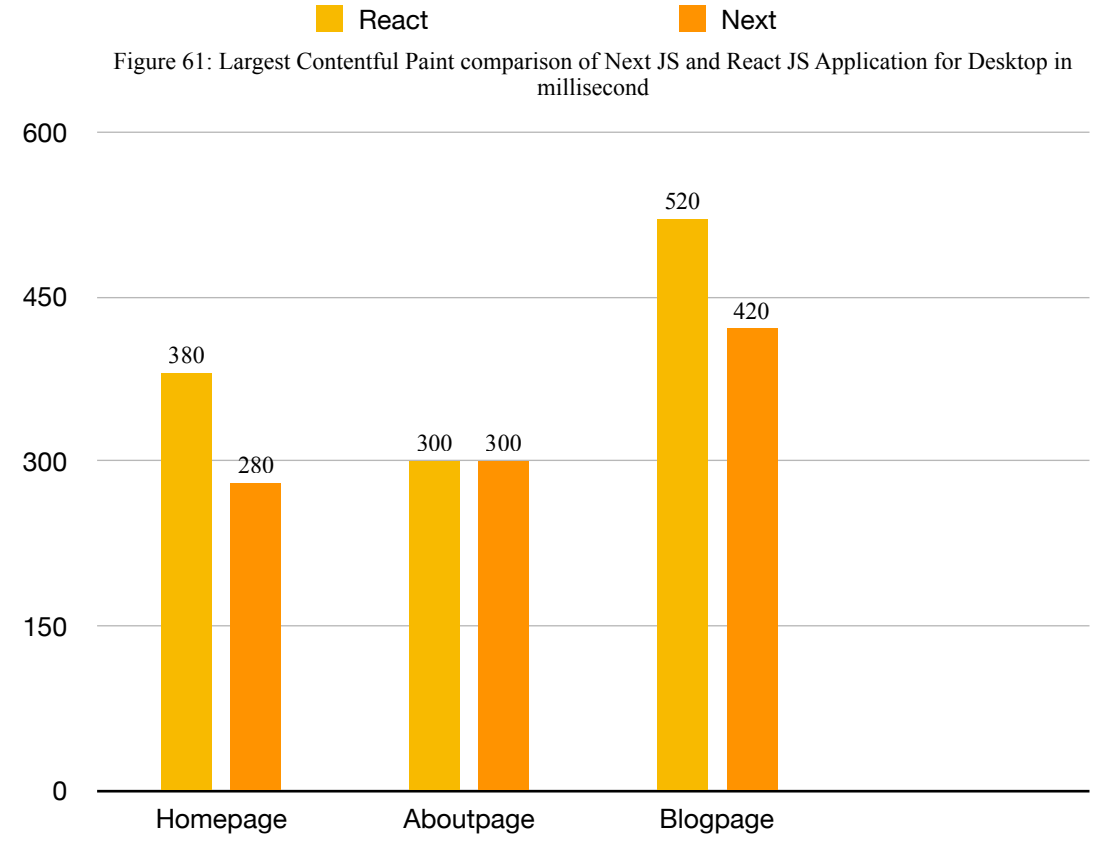


Figure 61: Largest Contentful Paint comparison of Next JS and React JS Application for Desktop in millisecond

5.3.3 Impact of Next JS on Total Blocking Time :

The Total Blocking Time (TBT) metric measures the total amount of time after First Contentful Paint (FCP) where the main thread was blocked for long enough to prevent input responsiveness. Total Blocking time can be reduced by reducing the impact of third party plugins, reducing the load from main thread, minimising the size of css and javascript & reducing unused css and javascript etc. Next Js application in our case is showing higher score than react application due to unused javascript .

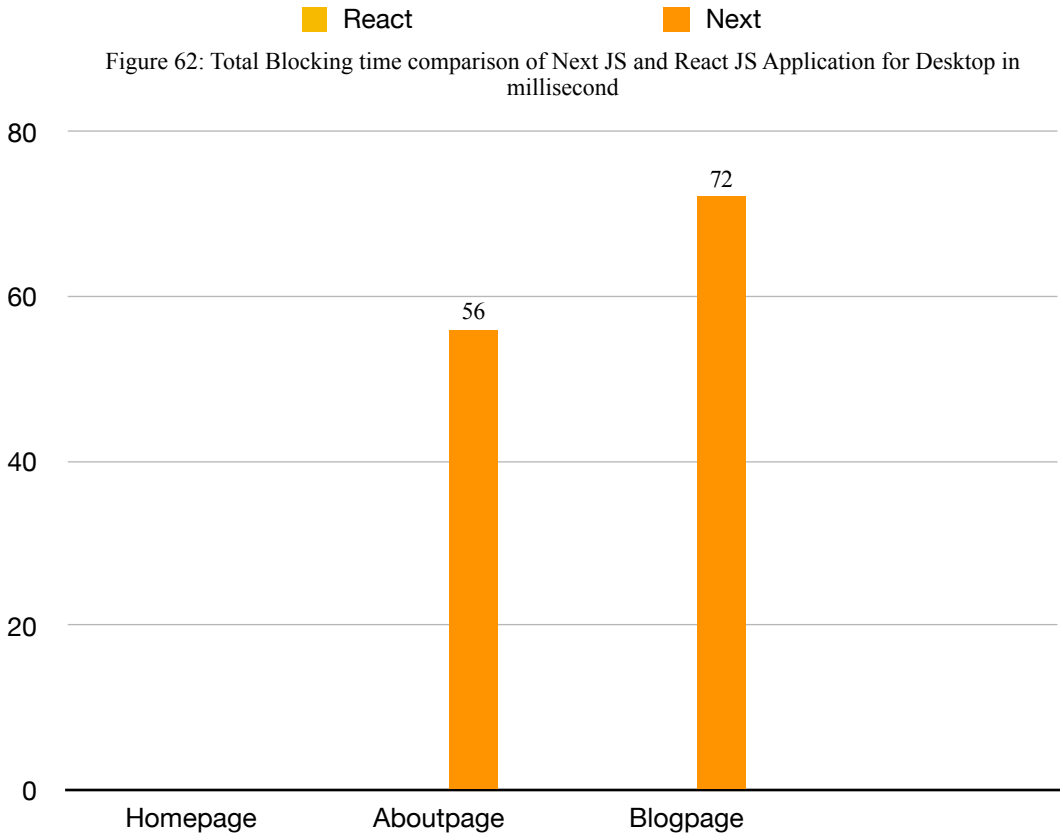


Figure 62: Total Blocking time comparison of Next JS and React JS Application for Desktop in millisecond

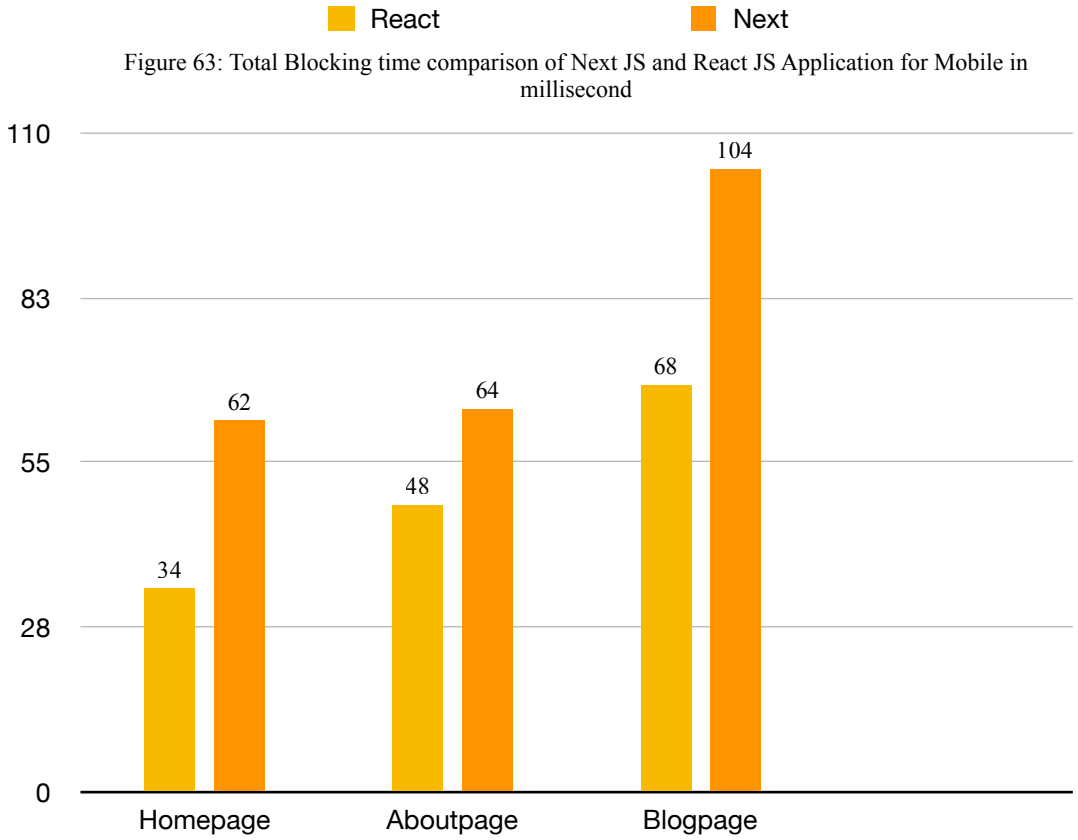


Figure 63: Total Blocking time comparison of Next JS and React JS Application for Mobile in millisecond

5.3.4 Impact of Next Js on Cumulative Layout Shift:

Cumulative Layout Shift measures unexpected layout shifts can disrupt the user experience in many ways, from causing them to lose their place while reading if the text moves suddenly, to making them click the wrong link or button. In some cases, this can do serious damage. Preloading the critical assets , fonts and script and optimising the images used on the application can improve the CLS score of our application . Next.js gives us advantage of next/images which incorporates several image-related best practices which is built on top of <img> HTML tag.

5.4 Overall impact of Next Js on the page performance of the application:

If we compare page speed score of the react and next version of the same application , we can see next application is having better Largest Contentful Paint and First Contentful Paint score than react application . Since our application is not having many folds so the Cumulative Layout Shift is good for both the versions and react application is having better Total Blocking Time than next application . If we consider the overall page speed we can see next application is having better score than react application . If we consider the loading speed of the application means the time it took to load the entire website, we will see next application is loading faster than react application , The visual complete of react application is 10, 000 ms where as for next application it is 4000 ms. Below we have highlighted different execution timings and visual progress of both version of the application.

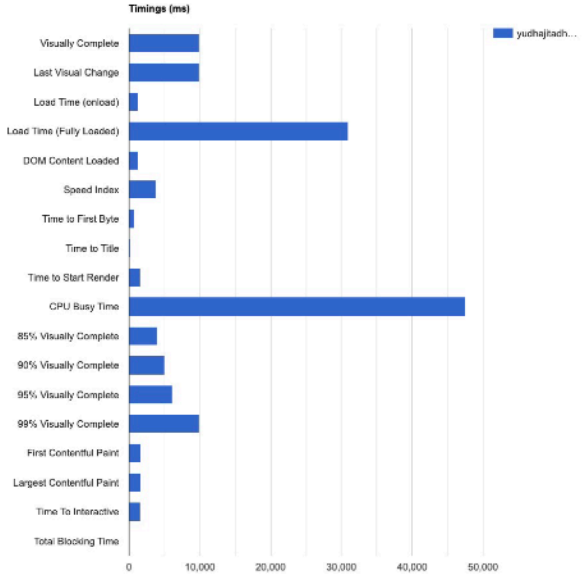


Figure 64 :Execution timings for React Application in milliseconds

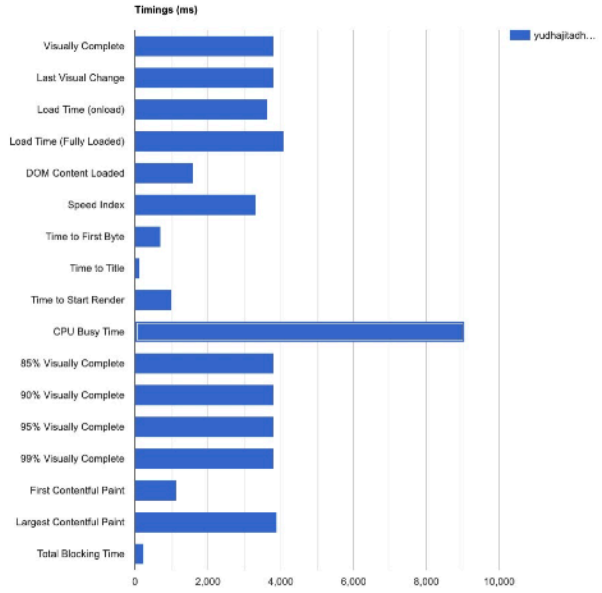


Figure 65: Execution timings for Next Application in milliseconds

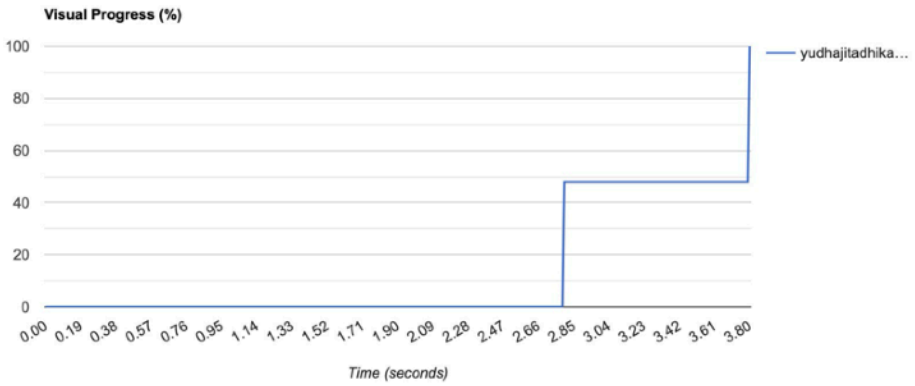


Figure 66: Visual Progress of Next Application

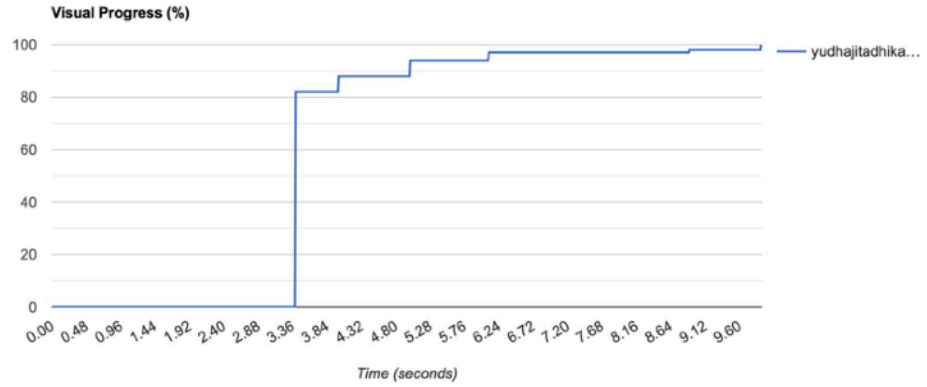


Figure 67: Visual Progress of React Application

If we consider bundle size of both versions of our application we can see image size of react application is 31,205,759 bytes where as next application is 2,947,457 bytes image size, but react application is having 57,009 bytes JS size where as for next application it is 93,572 bytes . Below we have highlighted different bundle size and overall pages speed of both version of the application.

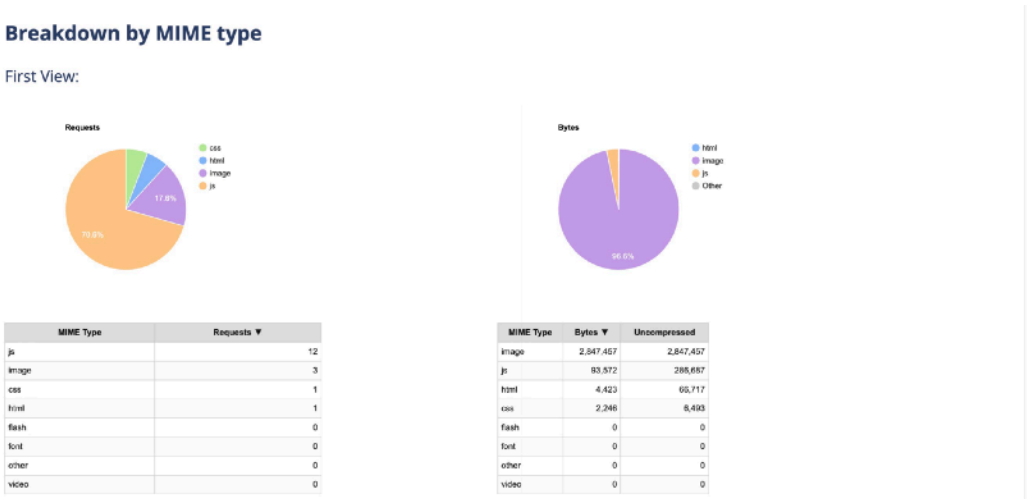


Figure 68: Bundle Size of Next Application

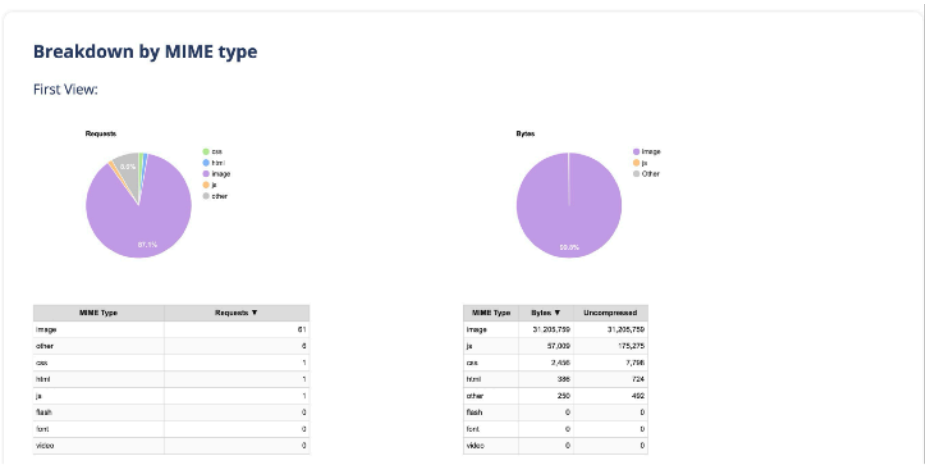


Figure 69: Bundle Size of React Application

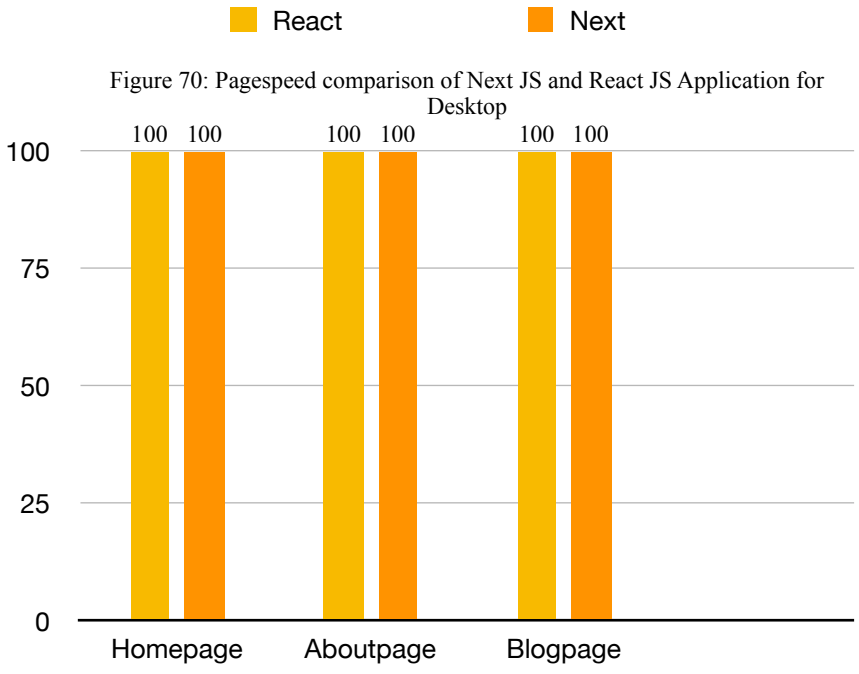


Figure 70: Pagespeed comparison of Next JS and React JS Application for Desktop

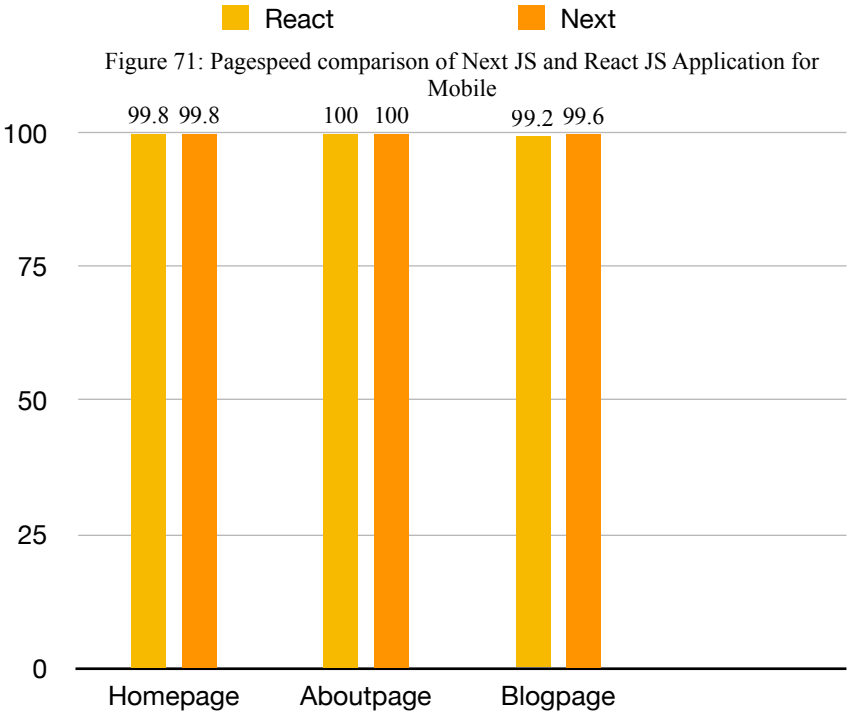


Figure 71: Pagespeed comparison of Next JS and React JS Application for Mobile

6. Conclusions and Recommendations:

6.1 Introduction:

This paper explored methods techniques to improve performance of modern web application using Next JS , thereby pointing out the factors affecting the pagespeed of our application and pointing out the optimisation features of Next Js . It compares the page rendering performances and the popularity of Next Js. Two similar web applications were built to evaluate the performance by using React JS and Next JS .

6.2 Discussion and Conclusion:

The method by which the performance comparison was conducted was by running pagespeed scan using pagespeed insights for both versions of the application. We also analyse diagnostics section of the page speed report and identify the issues which are affecting the page speed and recommendations provided to remediate those issues .The metrics which are considered are First Contentful Paint, Largest Contentful Paint, Total Blocking Time, and Cumulative Layout Shift. Except Cumulative Layout Shift, all metrics are speed measurements. The results from the comparison showed that Next.js performed better in most aspects. React.js performed better only in Total Blocking Time . So when our goal is developing a multipage scalable web app with tens of thousands of pages, it is all the more important to maintain a good balance between Next.js page load speed and optimal server load. Choosing the right rendering techniques is crucial in building a performant web app that won't waste hardware resources and generate additional costs.

6.3 Future Recommendations:

In conclusion, we can say there is no perfect configuration that suits all needs and purposes, and the best method often depends on the type of web application . However, we can start by determining the factors and picking the right Next.js rendering type and technique for your needs. There are some limitations to the results of this study, namely the availability of literature and time constraints to develop multiple applications for a better outcome. The impact of Next js on an application's performance will vary with it's complexity. Elevation of these challenges in the future may create a better understanding on optimisation techniques

of Next js. For example, developing large-scale applications and more complex applications might be necessary to reach a definitive conclusion . Loading test, simulating large numbers of concurrent users with multiple requests per second, is another thing to consider. Even with the mentioned shortcomings, this paper could be a starting point for deciding the speed optimisation techniques of Next JS.

Reference:

1. Zahra, Amalia, Handoko Wijaya Sukardjoh, Kristian, Khatib Sulaiman Dalam No, Jln (2023). *Website Optimization and Analysis on XYZ Website using Web Core Vital Rules*

2. Patel, Vishal (2023). *Analyzing the Impact of Next.JS on Site Performance and SEO*. International Journal of Computer Applications Technology and Research.

3. Tong, Jason, Jikson, Ricky Rivaldo , Gunawan, Alexander Agung Santoso (2023). *Comparative Performance Analysis of Javascript Frontend Web Frameworks*

4. Kynash, Yurii, Kolomoyets, Mykola (2023). *Front-End web development project architecture design*

5. Lyxell, Oskar (2023) . *Server-Side Rendering in React: When Does It Become Beneficial to Your Web Program?*

6. web.dev (2023). *Understanding the critical path.* Available at: https://web.dev/learn/performance/understanding-the-critical-path [Accessed : 4th March 2024]

7. *Sushin Pv (2022). NextJs Application Architecture for best performance.* Available at:*https://medium.com/@sushinpv/nextjs-application-architecture-for-best-performance-8f1d22e33ba1* [Accessed: 07th April 2024]

8. Wehner, Nikolas, Amir, Monisha, Seufert, Michael,Schatz, Raimund, Hobfeld, Tobias (2022). *A Vital Improvement? Relating Google's Core Web Vitals to Actual Web QoE*. 14th International Conference on Quality of Multimedia Experience, QoMEX 2022

9. Iida Kainu (2022). *OPTIMIZATION IN REACT.JS Methods, Tools, and Techniques to Improve Performance of Modern*

10. Sasikumar, S, Prabha, S,Mohan, B Chandra (2022). *Improving Performance Of Next.Js App And Testing It While Building A Badminton Based Web App*

11. Azeez, Abdulsamod, Ayodeji, Bolaji , Ayodeji, Bolaji (2022) *Performance and Accessibility Evaluation of University Websites in Nigeria*

12. Dinku, Zerihun (2022).*React.js vs. Next.js.* Metropolia University of Applied Sciences

13. Fariz M, Lazuardy S, Anggraini D (2022) . *Modern Front End Web Architectures with React.Js and Next.Js*

14. Harish A Jartarghar, Girish Rao Salanke, Ashok Kumar A.R, Sharvani G.S ,Shivakumar Dalali (2022).*React Apps with Server-Side Rendering: Next.js.*Department of Computer Science and Engineering, R.V College of Engineering, Bengaluru, India. Don Bosco Institute of Technology, Bengaluru, India.

15. Bryan McQuade (2022). *Defining the Core Web Vitals metrics thresholds* Available at: https://web.dev/articles/defining-core-web-vitals-thresholds [Accessed: 4th March 2024]

16. Desamsetti, Harshith ,Dekkati, Sreekanth (2021).*Getting Started Modern Web Development with Next.js: An Indispensable React Framework.*

17. Amar Sagoo , Annie Sullivan, Vivek Sekhar (2020). *The Science Behind Web Vitals* . Available at : https://blog.chromium.org/2020/05/the-science-behind-web-vitals.html . [Accessed: 1st March 2024]

18. Vasilije Vasilijević, Nenad Kojić, Natalija Vugdelija (2020) *A NEW APPROACH IN QUANTIFYING USER EXPERIENCE IN WEB-ORIENTED APPLICATIONS.* FOURTH INTERNATIONAL SCIENTIFIC CONFERENCE ITEMA 2020

19. Bangzhong Cao ,Minyong Shi , Chunfang Li (2017) .*The Solution of Web Font-end Performance Optimization*. CISP-BMEI 2017 : proceedings, 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics : 14-16 October 2017, Shanghai, China.

20. Mariko Kosaka (2018) , *Inside look at modern web browser (part 1) | Blog | Chrome for Developers.* Available at:https://developer.chrome.com/blog/inside-browser-part1 [Accessed : 2nd April 2024]

21. Ericsson (2016), *Streaming delays mentally taxing for smartphone users: Ericsson Mobility Report.* PRESS RELEASE FEB 17, 2016 09:00 (GMT +00:00) Available at : https://www.ericsson.com/en/press-releases/2016/2/streaming-delays-mentally-taxing-for-smartphone-users-ericsson-mobility-report [Accessed: 24th February 2024]

22. Kaaresoja, T., Brewster, S., & Lantz, V. (2014). *Towards the Temporally Perfect Virtual Button: Touch-Feedback Simultaneity and Perceived Quality in Mobile Touchscreen Press Interactions.* ACM Transactions on Applied Perception (TAP), 11(2), 1–25.

23. 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 (3)

24. Oulasvirta, A., Tamminen, S., Roto, V. & Kuorelahti, J. (2005). *Interaction in 4-Second Bursts: The Fragmented Nature of Attentional Resources in Mobile HCI.* In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 919–928).

25. Hoxmeier, J. A. & DiCesare, C. (2000). *System Response Time and User Satisfaction: An Experimental Study of Browser-based Applications.* AMCIS 2000 Proceedings, 347.

26. Card, S. K., Robertson, G. G., & Mackinlay, J. D. (1991). *The information visualizer, an information workspace*. In Proceedings of the SIGCHI Conference on Human factors in computing systems (pp. 181-186).

APPENDIX A: Research Proposal

Abstract

Website performance is the most critical aspect when it comes on creating a business.It has a huge impact on the experience user will be having , how much your website can convert user , what user thinks about your brand , website page speed have a huge impact on sales , activities and other services. User who have to wait for long time for site to load are more likely to have bad impression and that leads to leaving the site before performing the desired action. Good website performance create a positive experience which increases customer satisfaction which finally leads to loyalty and long term support. Economic Times achieved an overall 43% better bounce rate across the entire website by optimising the website performance. The number of visitor has improved by an average of 22 percent for Renault domains which is from 51% to 73%. With every second delay on their site , BBC have experienced lost of 10% of it's users. By improves the page-speed of their website, Rakuten 24' increased conversion rate by 33.13% and revenue per visitor gets increased by 52.37%. Vodafone found that improvement on their website page speed leads to 8% more sales , 15% uplifting in the visit rate, 11% uplift in the cart to visit rate . RedBus improved their website page speed which resulted increase in sale by 7%. So knowing the real cost of website that failing to engage, retaining users or not able to meet customer expectations makes us more curious about how we can follow few fundamentals for improving performance and SEO of the website. Next JS is a react framework for building web application, it under the hood , comes with many configuration toolings like bundling , compiling and more, so instead of spending time with configuration , developer can spend more time on building the application. In this research paper we will discuss how we can leverage the builtin optimisation features provided by Next JS to improve the performance of the website .

CHAPTER-1

INTRODUCTION

1.1 Background

Optimising the user experience quality leads to long term success for any website , whether you are in business, marketing or development.User using browser are on a journey , and their each action is like the steps, so like real world, delays are nothing but distraction from their activity, and this may leads them to make mistakes . So if our website is slow and non responsive it may lead to lower satisfaction and user will escape from the site or whole journey. A page loads more often when the user are keen to learn about something new like any recent affairs , a new product etc. So from user's prospective since they have not yet achieved their goals they may be less tolerant to the website delay. The effect of website delay varies and it's hugely depends on user characteristics , past experience and the task urgency . According to Galletta, D. F., Henry, R., McCoy, S. & Polak, P. (2004) delay that can decrease satisfaction and intention to return on the any random site is 2 secs . According to Hoxmeier, J. A. & DiCesare, C. (2000) if we delay loading each panel navigation from 0 to 3 seconds satisfaction drops and when it drops from 9 to 12 seconds the intension to return also dropped. So we can say 6 seconds delay is enough to declare a website as slow. Oulasvirta, A., Tamminen, S., Roto, V. & Kuorelahti, J. (2005) says that mobile user didn't keep their attention on the screen for more than 4-8 seconds, so we can say a 5 second delay will seem 10 sec delay for a mobile user .

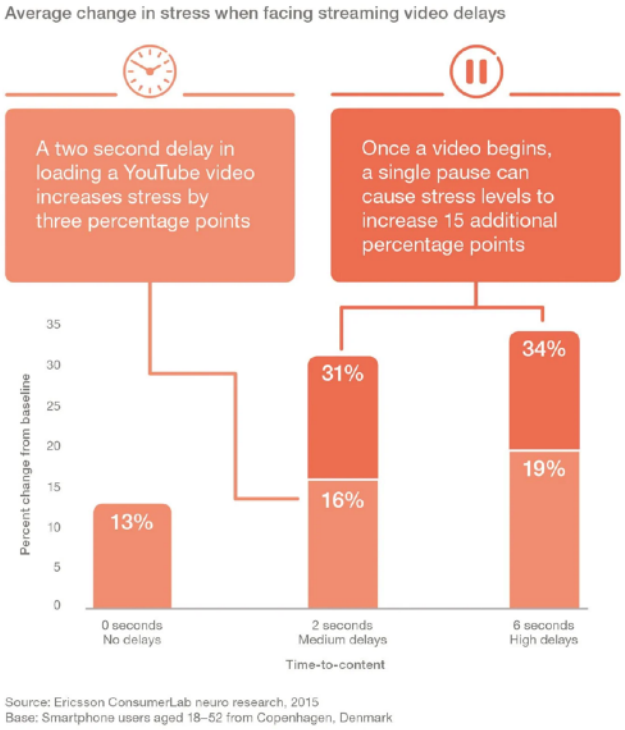


Figure 1: Increase of stress level of users due to delay of two second. (Ericsson 2016)

Card, S. K., Robertson, G. G., & Mackinlay, J. D. (1991) suggested that the speed of a system's response should be comparable to the delay human's experience when they interact with another, a response should take around 1- 4 seconds .

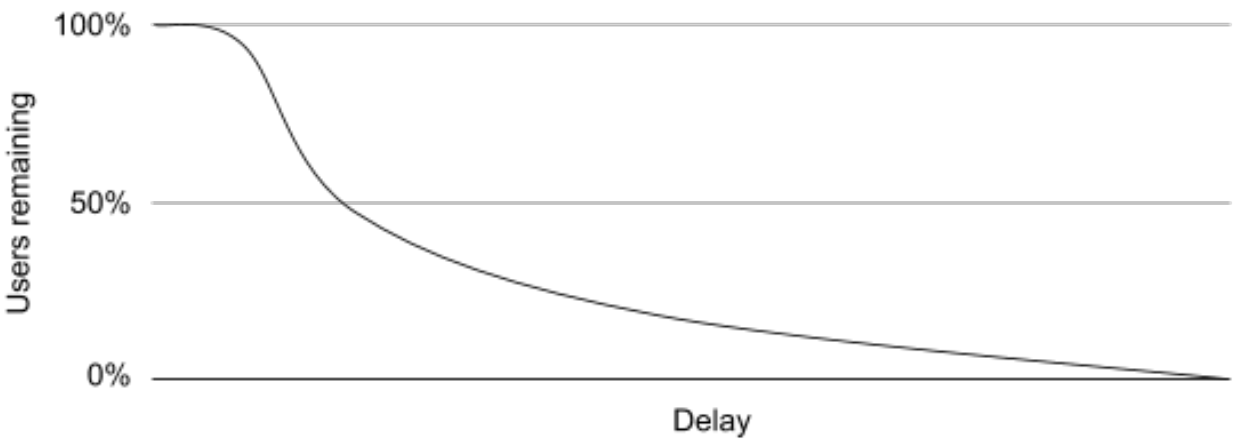


Figure 2: Graph of website delay vs users escaping website(Amar Sagoo , Annie Sullivan, Vivek Sekhar 2020)

An application speed depends on how long it takes to provide all the website related content and data to the client in first circle. Application execution collapses when server sends some render blocking resource during this circle . In this paper we will implement and analyse how we can use Next JS to improve the performance and search engine optimisation of a website by optimising those blocking resources.We will also discuss how we can implement those Next JS features practically in our code to improve the performance of a website. In this research paper, we will be analysing how Next JS can improve the page-speed of a website. For that first we will be analysing what are the factors that affects the page-speed of a website .Then we will explore and analyse how Next JS can leverage the page-speed of a website. After that we will be conducting a page-speed scan on a reference static site using Google page-speed insights tool , it is a tool used for improving the quality of web pages and is able to run a variety of test against a webpage while monitoring various performance like the core web vitals and speed index. Then after analysing the scan report we will be developing the same application using Next JS and will conduct another page-speed scan on that application . Finally we will compare and evaluate both the page speed scan report and will try to draw a conclusion on how Next JS is impacting the page-speed of the application .

1.2 Problem Statement

Web performance is a very critical aspect of web development, by optimising our website we are giving our user a better experience which help us in increasing traffics of our website and finally leads us to business improvement. Slow website will always have a negative impact on revenue , where as a fast site will always increase conversion rate and improves business outcomes . According to Ericsson(2016) the amount of stressed people gets while waiting for the delay in page speed is equivalent to that of watching a horror movies or solving a mathematical problem and greater than waiting in a checkout line at a retailer.

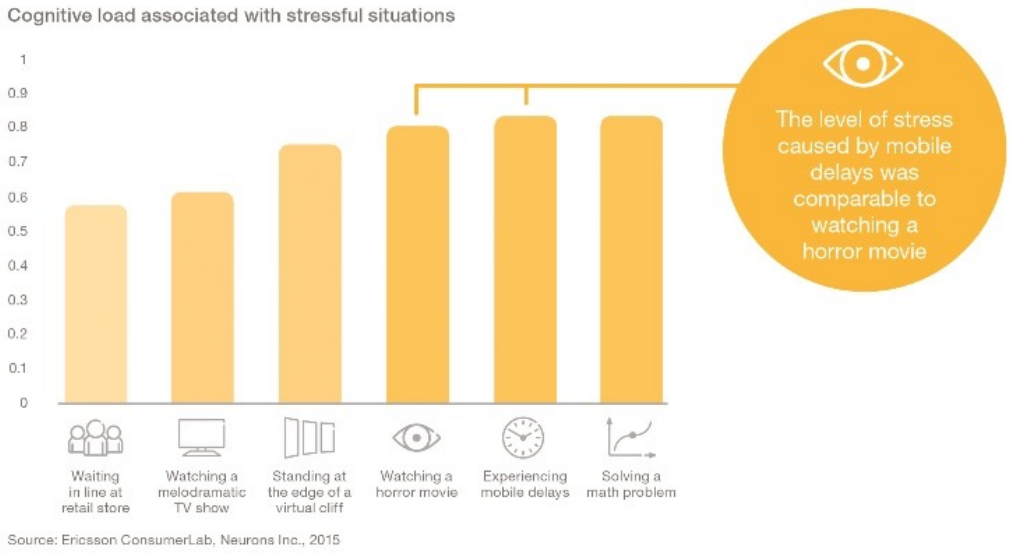


Figure 3: The level of stress caused by mobile delay (Ericsson , 2016)

When a user enter a url in the browser it sends GET request to the server to fetch it ,the HTML resource is requested first for a web page , so to ensure a good page speed of our website we should always ensure the HTML arrives quickly with minimal delay. Mainly there are two types of resource that blocks a HTML to arrive quickly, render blocking resource and parser blocking resource . In this paper we will discuss and analyse how Next JS , a react js framework efficiently handle those resources by applying it's builtin optimisation techniques , and how we can leverage those features technically to improve website performance.

1.3 Research Question/Aims and Objectives

In current scenario , maintaining a good page speed of a website across all browsers and device are very important to stay ahead in business competition , through website we used to showcase our products , our business ethics , brand value and loyalty to our customer , whether we are in business , marketing or development optimising a good page speed will always keep our website and our business ahead of others. The main aim of this research is to evaluate the effectiveness of Next js for improving performance of the website. The objectives or the steps to achieve the aim of the study are as follows:

a> To identifying factors which are responsible for slowing down the rendering process of the website .

b> To identify the optimising properties of Next JS which can help to optimise the performance of the website.

c> To draw conclusion based on the findings of the study.

1.4 Significance /Scope of the Study:

As modern web application grows , it becomes increasingly difficult to maintain their performance , that is why it is very essential to optimize application performance, by optimising our website we are giving our user a better experience which will help us with increasing traffic and finally leads to business improvement. Now let's discuss the scope of the research paper.

a> In this research paper we will be developing a real world application (https:// yudhajitadhikary.github.io/Portfolio) using React JS .

b> We will be generate performance report of React Application using google page-speed insights.

c> We will be measuring the page speed of the site using metrics defined by google core web vitals.

d> We will go through the report and will identify the loop holes present on the site which are affecting the page speed and will go through recommendation provided by google for improving the page-speed of the application.

e> We will developing the same application using Next JS and will try to apply those recommendation using Next JS.

f> Finally after applying the Next optimisation strategy on the application we will be generating the performance report of the optimised website for comparing and analysing the impact of Next JS on improving the page-speed of the website.

In this paper we will evaluating our website performance using google pagespeed insights, and will follow the core web vitals standards and recommendations provided by google.We will also consider the metrics defined in core web vitals as performance measuring parameters of the website . This paper will be written based on the pagespeed report generated in google page speed insights.

1.5 System Architecture Design:

The Layout of the next application can be divided into three sections:

1. Network Layer
2. Next Application instance: the SSR
3. External Service

1. **Network Layer:**

Network layer used to decide how the request will be handled by it's server and which protocol will be suitable to be utilised , When an application is requested by user, first the request goes to load balancer, load balancer check the request if the request is for loading website HTML content , it will send the request to the application server , else if the request is for loading static content like images, fonts , or a build file it will send the request to CDN and content will load from the CDN by this way we can reduce the static request load on the application server using load balancer to deliver all the static content from the edge location.

2. **Next Application instance: the SSR:**

This section determines how the application code should work , all request go through the next js custom cache server , which used to cache all incoming requests. When a request comes for the page mostly in server side rendering the next application loads all of the material and generates a HTML version of the page request. If we want dynamic data to be loaded with the HTML we can use Next Js implementation like getServerStaticProps, getInitialProps or getStaticProps. NextJs comes with Next Image which is a loader that transcript the image into a certain size or format to minimise the time of transmission over network , we can use webp format image and avif images as well to optimise the image size of our application.If we want to serve dynamic static content or if we need to store a sitemap that is very dynamic we need to store it alongside the website itself , we can use file request for dynamic routing .

3. External Service:

When user enter the website url on browser , it sends the first initial page request to the server , When load balancer receives the request it will route the request to custom server provided by Next application. It will check whether the file is already been cached or not , if it's cached , it will return the cache data, and if it is not cached ,it will load the next application and will run the server executable like GetStaticProps , GetServerSideProps and GetInitialProps to loads the complete next application and convert it to HTML . Then it caches the HTML file that has been parsed , and return the data to the client .Then the HTML markup is downloaded and starts loading in the browser , while loading it sends a request to render all the essentials like Javascript , Fonts , Images , CSS etc to load balancer. Load balancer will send the request to CDN and static storage bucket , we can also create a sub domain and connect the CDN to storage bucket directly bypassing the application load balancer. The website will initiate loading in the client side while all of them have been loaded , and if any frontend api needs dynamic loading , parallelly it can perform the calling and storing the data in state management system like redux or mobx. Now last step is loading the next page, when user moves from page A to page B , the Javascript required to load Page B will be preloaded and api request to get the content like getStaticProps, getInitialProps and getServerSideProps function will be initiated. If it is successful, the page B will start displaying with all the scripts and content to make it interactive.

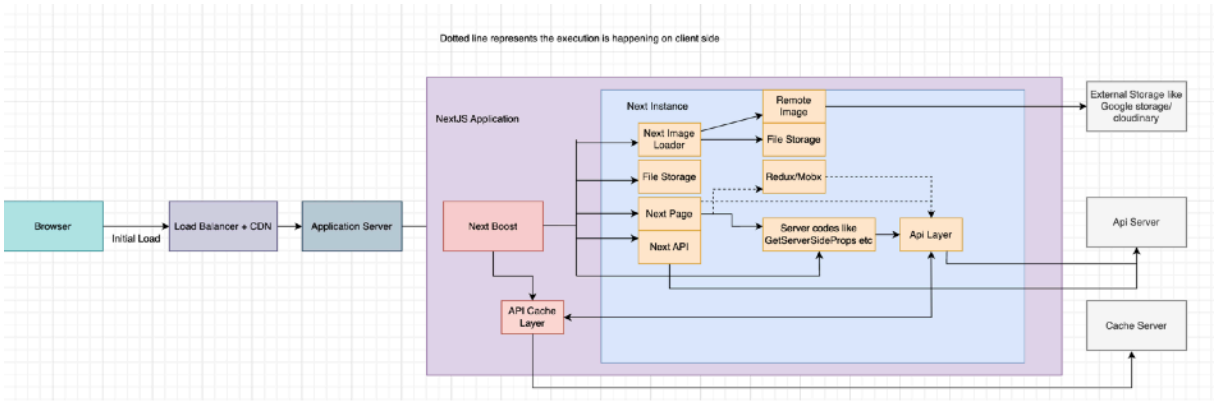


Figure 4: Next.js system architecture design

According to Sushin Pv(2022) the major reason for utilising Next JS is to have a default support for SEO , we can use Next JS script and can use different strategy to load the required third party scripts , while developing a page we should always keep our eyes on the build size , loading properly sized image and prefetching them impacts the page speed of the site a lot , we can use next image , it is a lazy loading technique to load the image , it is also vital to load images which are on the first fold of the website , like if we have 300*300 px in canvas , downloading 1200*1200 px will be waste of internet bandwidth.We can use dynamic import for lazy import which will prevent all unnecessary code from being bundled together. Next JS preloads all the JavaScript for pages having a link tag to them from current page , this speeds up the client side rendering , but if we are having a link in footer , user more likely will not click on those links , so we can mention prefetch=false on those link tags to prevent them from been preloaded . Next Js also provides builtin support for AMP, which increases page speed by creating high value page to AMP and make them load more faster.

CHAPTER-3

RESEARCH METHODOLOGY

3.1 Importance of Performance

The success of online venture is highly dependent on website performance, retaining user is very important for improving the customer conversion , website that respond to the user interaction and inputs quicker and in proper way can retain and engage user more than the website which is non responsive or have a slow response time, performance can also have a material effect on whether your website will fall through, performance is not just about business status , when it comes to user experience speed matter , As a page loads there is no user experience or interaction in the period when the site is loading , at that interval user are actually waiting for the content to come on the site , this lack of experience is fleeting and facilities customer towards lower commitments since user are forced to wait . Website comes with many images , third party script and lots of code , browser needs to download those assets for rendering those assets in the site , which need few megabytes of users data plan, specially for mobile device , with limited CPU power and memory downloading those assets can create poor performance conditions. In order to give good experience to the user performance is very important aspect. When it comes to investors and stakeholder they should always value the importance of website performance , because it comes with all other aspects like user centric performance metrics, customer satisfaction which finally leads to business growth and improvement .The way to achieve a excellent performing website depends on where our website is present in current scenario from performance prospective , what is the user complexity , design complexity , what are the easy fix we can apply to the website which will show a significant impact on performance growth of the application .By delivering a good performing , responsive and quick interactive website to our users we are giving our customer a seamless delightful experience , which make them our loyal and returning customer .With the advent of framework like Next JS we can optimize our website very easily .

3.2 Critical Rendering Path

Optimisation critical rendering path is essential for improving a web application. The critical rendering path follow different steps which are as follows:

a> It first construct the Document Object Model (DOM) from the HTML received.

b> It then construct the CSS Object Model (CSSOM) from the css file it received.

c> It use to identify and apply the Javascript that can alter the document object model and CSS object model.

d> It used to construct the render tree with the DOM and CSSOM created .

e> It use to perform layout and style operation on the page to observe which element will fit in which part of the DOM.

f> It paints the pixels of the element in memory and compress the pixels if any of them are overlapping.

g> Finally it draws all the resulted pixel to the screen of the browser.

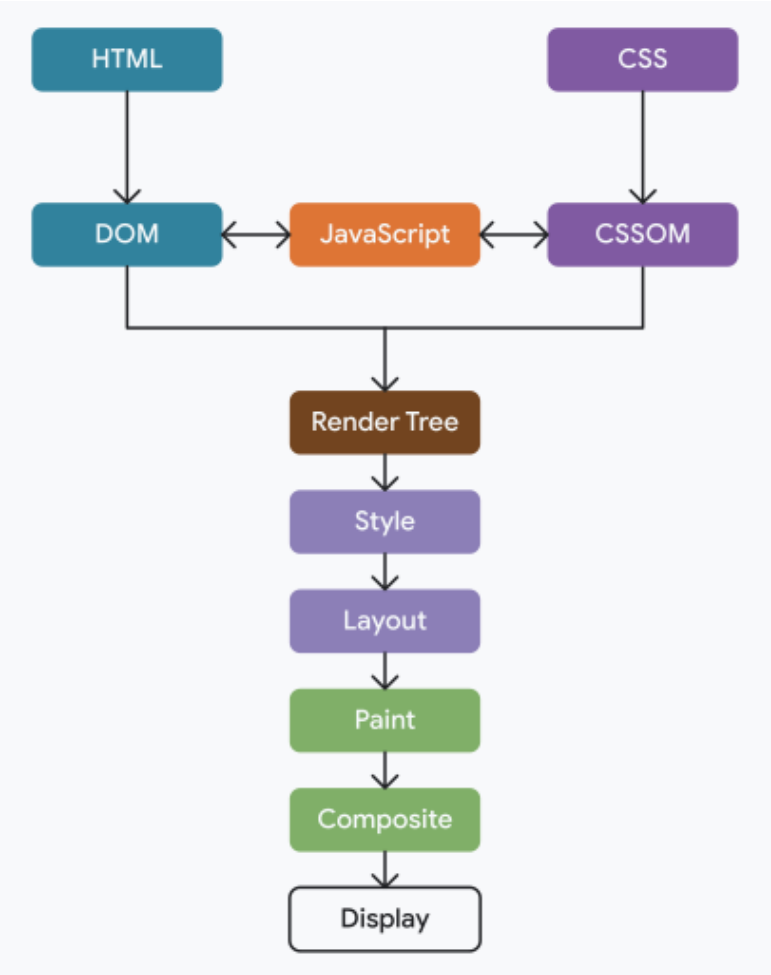


Figure 5: Critical Rendering Path (web.dev, 2023)

3.3 Advent of V8 to redefine JavaScript

Google created the Chrome V8 engine in 2008, before the introduction of V8 the main job of Javascript was to communicate with css for building a user interface and it used to execute few common function like validation etc. With the advent of chrome V8 engine google pushed JavaScript to the forefront of the HTML5 . According to Jartarghar HRao, Salanke, GKumar , ADalali (2022) V8 JavaScript actually redefines the JavaScript which increased the speed 56 times quicker than any version of Internet Explorer (IE).Traditional web browser use to create JavaScript by parsing byte code and compile whole web project to generate the code and once JS is created it will get complied from file system . As a result JavaScript execution is a big blocker and is significantly longer than any complied language like Java and C++. Because of the excellence of V8 engine, different JS platform based on V8 engine started emerging which brings a new era on website development .Node Js was introduced on 2009 , with Node js developer started understanding that JavaScript can do more than just running a simple script on a webpage . Now let's discuss two important concepts that solidifies the concept of web development and render process ,Client side rendering and Server side rendering .

3.4 Client Side Rendering VS Server Side Rendering

3.4.1 Client Side Rendering

Client side rendering uses JavaScript to render the content in browser , as a result initially few part of HTML document loads with Javascript, rather than waiting to receive all the page content directly from HTML document , the remainder of the site is then rendered using the browser .Initially the page load used to be slow with CSR but sequentially all the pages loads very fast . In Client side rendering server obtains data at run time and each call does not requires complete User interface reload , it only used to refine the DOM element with the updated data . Famous framework like React JS , Angular JS , Vue JS uses Client side rendering .

3.4.2 Server Side Rendering

In Server side rendering server send the whole HTML response to the browser over the internet , it is browser responsibility to render the page , server use to do entire process like requesting data from the database , creating the HTML and sending it to browser to render etc . Next Js uses Server Side Rendering .

3.4.3 Client Side Rendering Vs Server Side Rendering

According to Jartarghar HRao, Salanke, GKumar , ADalali (2022) the main difference between client side rendering and server side rendering is that in server side rendering server sends ready to render HTML of an application to the browser, but in client side rendering browser will just get a sparse of document with a link to Javascript.In both the situation, it will go through the step like creating a virtual DOM and adding event to make the page interactive . In server side rendering the user can see the page from beginning while all the process are happening . In client side rendering the same process is followed before the virtual DOM is sent to the browser .Web application when use server side render approach loads quickly , where as when it loads with client side rendering it shows blank, white page initially. Before going into the details about web performance , we should first understand the render process of a website , because to understand how to optimize a website we should be aware of how website rendering process happens.

3.5 Process of rendering Website

Many activities or process used to happen under the hood on the process of rendering a website . Renderer process is responsible for everything that is happening inside a browser tab . Main thread mostly have to parse most of the code we send to the server.There are two other threads called compositor and raster thread which helps to make the render process efficient and smooth. According to Mariko Kosaka (2018) the main job of renderer process is to convert JS ,CSS and HTML into a webpage that can interact with the user . Now let us go through each of the rendering process in details.

3.5.1 First step is construction of a Dom :

At first the navigation tab used to send commit message to renderer process , to start receive data of HTML , then HTML is parsed by the main thread which is defined by the HTML standards. The internal structure of a page and data of the browser by which web developer can communicate with via JS is called DOM.

3.5.2 Sub resource Loading:

A website usually have many extra resource like JS , CSS , image etc, these resource needs to be downloaded from network or cache to build the page . The main thread use to request them one by one , but in order to speed up this process preload scanner is used , preload scanner used to preload <image > and <link> tags by peeking at the token generated by HTML parser and sending request to the network thread.

3.5.3 JS can block the parsing:

When the HTML parser encounters any script tag , it used to give it more priority by pausing the whole parsing process of HTML and start executing , parsing and loading the JavaScript first because it can change the structure of the DOM . That's why the HTML parser have to parse the JS script first before it can continue parsing the HTML document .

3.5.4 Hint to browser how you want to load resources :

There is a way by which we can give hint to the browser in which order they can order the resource , we can add async or defer attribute to the script tag , and give hint to the browser to load and run script asynchronously , without blocking the parsing. We can also use <link rel="preload"> tag to inform the browser that the resource is definitely needed for website navigation , and it needs to be loaded as priority.

3.5.5 Style calculation:

The main thread also parse the CSS and determines the computed style of each DOM node , it used to define what style will get applied to each of the element of the DOM structure based on CSS selector. Each DOM have a computed styles like <h1> is bigger than <h2> etc. We can check the computed style of the components of a website in the computed style tab in dev tool.

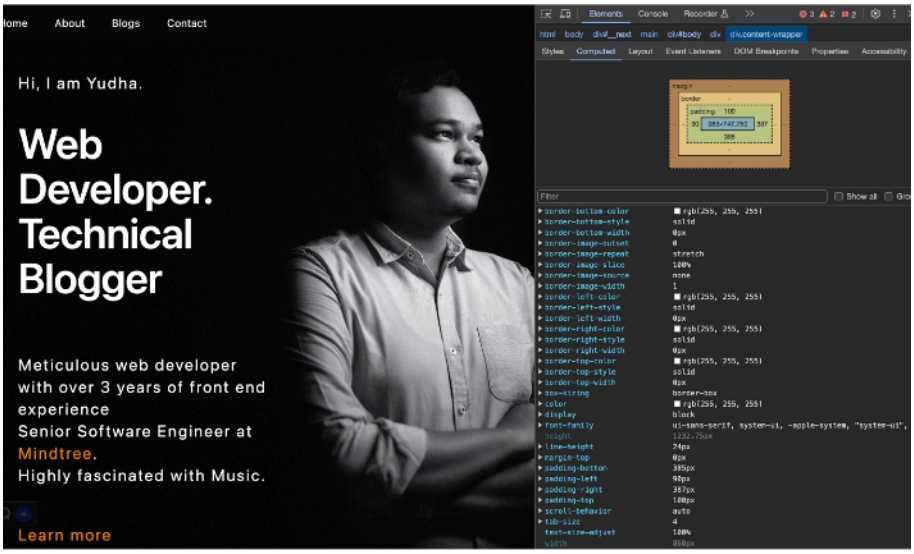


Figure 6: Computed styles of the application

Since we have created the DOM and computed style , now let's create the geometry of the elements to build page .

3.5.6 Layout :

Layout process is the process of designing the geometry of elements, Layout is similar to DOM tree , only Layout contains the data of the visible component on the page .Main thread creates the layout which will have information about the x and y co-ordinate of the element with boundary box size by crawling through computed style and DOM .

3.5.7 Painting :

So now we are having the DOM , computed style and layout of the website , in order to reproduce the painting of the website we need the location and order of the element. One interesting point of renderer process is in each step the result of the previous operation is used to create new update , so if there is any essential change in the layout tree , the past order also needs to be regenerated for the affected part of the document . Usually browser

use to run this operation in every frame (60 times in a second). Even if the renderer operator keeps reloading the document and refresh it in every frame. The calculation happens in the main thread only, which may block the application when Javascript is executed. For this reason it used to cluster the JavaScript into smaller chunk and schedule those to run at each frame. Now the browser knows the structure of the document , the style of each element and the proper ordering of the website elements. Turning these element into pixel on a screen is called rasterising. Previously chrome used to raster when user scrolls the page . Modern web browser used a sophisticated concept called compositing , which separates the part of the page into layers, then raster them separately and composite a page in a separate thread called composition thread.

3.5.8 Dividing the layer:

Main thread used to walk through the layout to create layer and try to find out which element needs to be in which layer.

3.5.9 Raster a composition of the master thread:

Once the layer is created and proper order are determined . The main thread sends those details to composition thread , after receiving the details composition thread starts rasterising each layer . The composition thread used to prioritise differ raster thread so that the elements which are coming within the viewport rastered first . Once tiles are rastered, composition thread used to gather tile information and store in draw quad or composition frame .

Draw Quad: It contains information such as the tile's location in memory and when is the page to draw the tile taking in consideration of the page compositing.

Composition frame: A collection of draw quad represents a frame of reference of a page .

A composition frame is then submitted to the browser processor via IPC. And then composition frame are stored in GPU to display it on a screen.The benefit of composition is that it is done without involving the main thread, composition thread does not need to wait for the style calculation or JavaScript execution.

3.6 Factors which affects the web performance :

Web performance is a very critical aspect of web development, by optimising our website we are giving our user a better experience which help us in increasing traffics of our website and finally leads us to business improvement. Keeping this in mind Google have initiated Core Web vitals to provide unified guidance for quality signals that are essential for delivering a great user experience on the web. In the current scenario consumers increasingly rely the web to access digital content and service, while they are browsing through your website they are actually comparing you and rating you with your competitors against the best-in-class service they use every day .So overall we can say performance is a foundational aspect of good user experiences. We will also try to find out what are the factors and features which can affect the page speed of website. Native web-app typically have a initial phase and running phase , however unlike native applications for web page and web application the lines between these two phase are much less . Once browser have resource to render a page the browser decides when to render and when it is too early, if browser render as soon as it just have some components with css and js , the page can momentarily look broken and on the other hand if browser wait for all new resources to be available instead of doing parallel rendering the user will be left waiting for long time , so sequence of steps the browser takes before performing initial render is very critical and is know as critical rendering path , when it comes to webpage performance improvement and optimisation minimising critical rendering path plays a vital role.

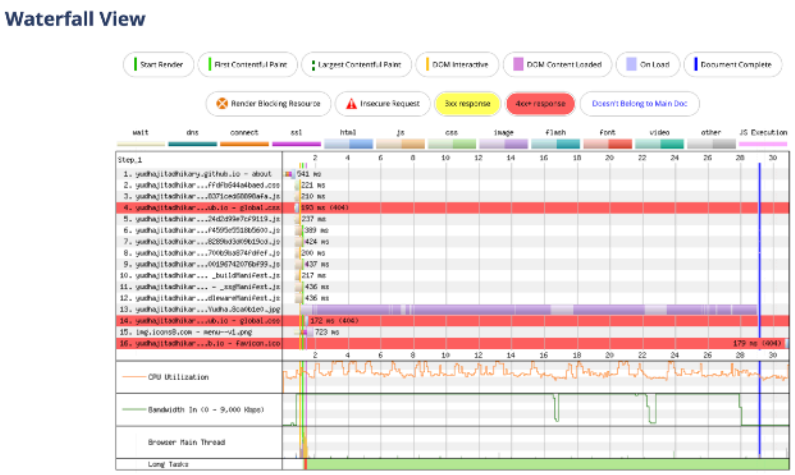


Figure 7: Waterfall view of https://yudhajitadhikary.github.io/Portfolio/about

Some resource are very critical for which browser pauses entire rendering until it dealt with them. Browser try to be as efficient as possible whenever it sees css resources it pauses rendering and will start processing rest of the HTML ,so it will block rendering , so such resource (mainly css) are know as render blocking resource .Another resources are parser blocking resources these are the resources that prevent the browser from looking for other work by continuing parsing the HTML,Mainly Javascript by default is considered as parser blocking resources. In this paper we will be analysing how we can optimize those resource using Next JS strategies and we will discuss how Next js , a popular React framework , can help us improving the performance of our website.

According to Bangzhong Cao ,Minyong Shi , Chunfang Li (2017) the steps followed to render a webpage are as follows :

1> When user enter the access url in the address bar of the browser.

2> The browser checks for the IP address in the domain name resolution library.

3> The client browser sends the request url to the server.

4> When server receives request to generate HTML from the browser , the server then sends response to the browser request.

5> Browser accepts a parsed data send by the server , server also returns the related resource file to the browser.

6> Finally browser will be parsing the related resource file and render the page from top to bottom.

Now let's try to figure out the waiting time involved in this whole process:

1> The time when server are resolving domain for the server IP also know as domain resolution time .

2> Time required by a client to establish the connection with the server.

3> Time taken by the TCP to establish a connection .

4> Time server takes to respond to the HTTP request.

5> Time required to download the request content after the server respond to the request.

So the most common factor to slow down page speed on a web page includes scripts, fonts , plugins (html, Javascript and css).

3.7 Current trends of Next JS

In 2024, Next.js development trends include hybrid rendering for optimal performance, a focus on accessibility and inclusivity, and enhancements to the developer experience with features like built-in TypeScript support and advanced debugging capabilities. To verify the current popularity of Next JS , we will go through few surveys recently done on few well known websites and platforms .

3.7.1 Google trends:

To understand the recent status of popularity of Next Js, we can check the google trends portal , google trends is a portal where we can compare the interest over time graph of several keyword or topics . When we compare Next JS with few other frontend development packages like vue.js , nuxt.js .We can see that Next js is having a very consistent and highest interest/search index , than other two frontend packages .

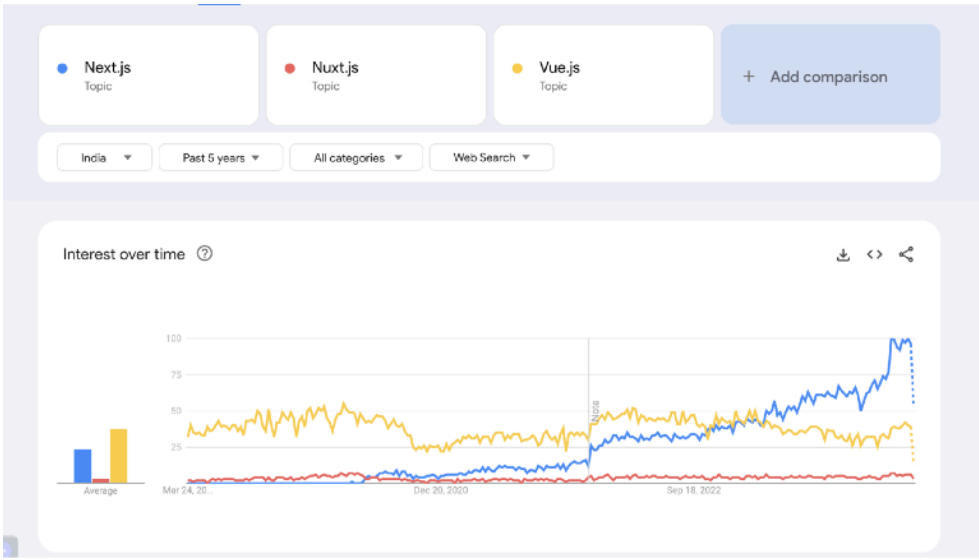


Figure 8: Interest over time on Google trends

3.7.2 Npm trends:

Npm trends is website to compare packages download statistics , GitHub stars , ratings etc . When we compare our frontend packages (nuxt js and vue js) with Next JS with respect to number of downloads in past 1 year. We can see that Next js is having the highest number of download in past 1 year.

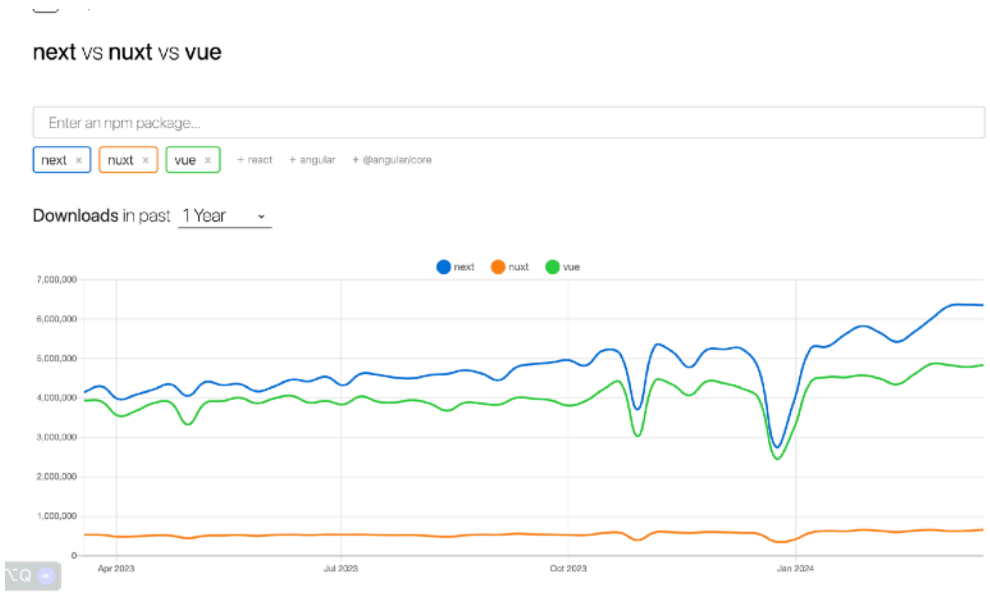


Figure 9: Npm download record in past 1 year for Next JS

3.7.3 Stack Overflow:

Stack Overflow is the largest, most trusted online community for developers to learn, share their programming knowledge, and build their careers. We will check survey result of the most popular web framework and technologies, the respondents of these survey include of coding learners , professional developers , architects etc . So according to the report Node.js and React.JS are the two most common web technologies used by all respondents. But point to highlight from these survey is Next.js moved from 11th place in 2022 to 6th this year mostly driven by it’s popularity and optimisation strategies.

Figure 10: Most Popular technologies or framework

Now we will check how web frameworks have trended over time based on the use of their tags since 2008 in stack overflow to analyse overall trends of frontend frameworks , there also we can see React JS is highest in recent trending and as time passed Next Js over performed and become second highest trends in current years .

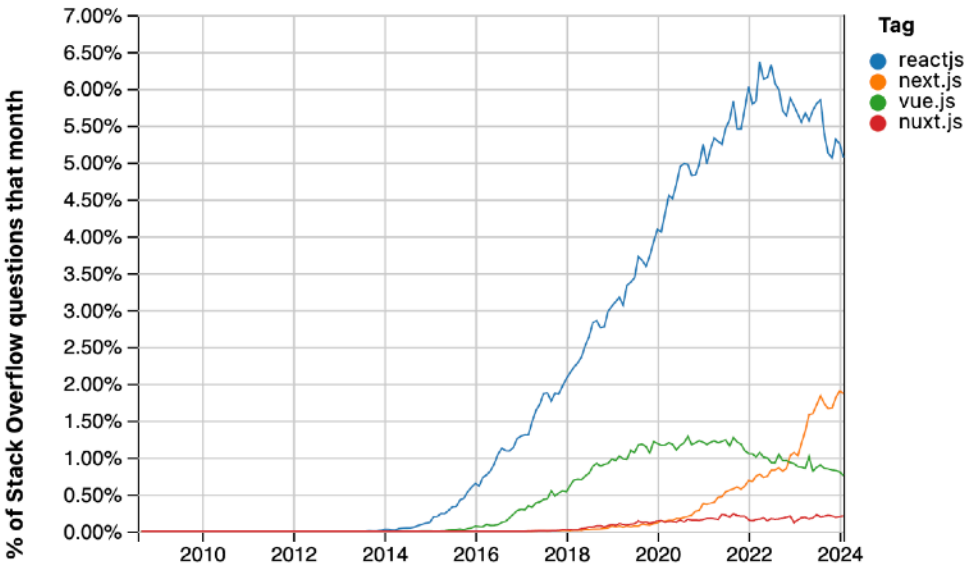


Figure 11: Overall trends of frontend framework

So we can seeing a good amount of growth on the usage of Next Js in current years , because Next js incorporates developer experience with production ready features like , server side rendering , typescript capability , smart bundling , route prefetching feature , optimised format of image , lazy loading , dynamic routing , file based routing and dynamic import etc.

3.8 Optimisation features of Next JS

Next JS provides many builtin features like file based redirection and routing, server side rendering (SSR), static side generation (SSG) which ensure improvement in performance of your application.Next JS also provide functionality like automatic static optimisation, image optimisation, code splitting , font optimisation ,optimising the dependency script and prefetching.

3.8.1. Automatic Static Optimisation:

Next JS also render our application automatically as static html , if our application is not having dynamic routes. If neither getInitialProps nor ServerSideProps are defined , Next JS will not re render the page for every request , this improves the performance by decreasing static html generating time of the application . If page are having dynamic routes using Next js getServerSideProps and getInitialProps we can pre-render the pages on the server and fetch the necessary data before sending the HTML to the client , which leads to improvement in performance of the page .

3.8.2. Code Splitting:

The process of segregating our code into smaller chunks is called code splitting. Next JS automatically segregate our code into maintainable chunks so that it takes less time to load. We can use dynamic rendering of the component provided by Next JS which will load component whenever it comes on screen, this will help us to further utilise the benefits of code segregation.

3.8.3. Image Optimisation:

Next JS is having a builtin feature called next/image component which automatically resizing or compression image based on different device size. The process of loading image whenever needed is called Lazy loading , this will load image when they are visible on the screen . Automatic image optimisation helps to reduce the time to load the image in our application . Using Next/image we can load immediately the image which are on the 1st fold by setting the priority property on the next/image as true. Next JS will prioritise loading the essential images we want to load first .

3.8.4. Font Optimisation:

Next JS also use the next/font component to optimise fonts, it will automatically download the fonts on server which will reduce the load time the fonts in our application.

3.8.5. Prefetching Facility:

Next JS is having Link component from next/link, the best feature of next/link is it automatically prefetch the code of the page which user may like to visit in next stage.

3.8.6. Optimising Dependency Script:

Next Js is also having next/script component which allows browser to first load the critical component , then the script . We can also ensure the script will run once , which is highly useful in the scenario where a common script is required to be loaded all across the site .

3.8.7. Critical Rendering Path Optimisation :

Next Js also helps to reduce the critical rendering path, it focus on the process for initial rendering of our webpage In this whole process browser needs to wait for some critical and essential resources to download needs before it can start with initial render ,those are part of the HTML , render block css and JavaScript script in the head element.There are few ways by which we can improve LCP of our application some of them are minimising render blocking resource like css and js , minimising the main thread work . Application main thread parses the HTML , executes JavaScript ,render the pages , so we should target minimising the work assigned to the main thread that will improve the LCP of the application , Javascript is single threaded language , which means at a time one task can be executed , so if we target reduce the JS executed on main ,thread we will get a good amount of impact on initial load time. We can use async function and Next Js also provided a component called next/script to load script , which allows browser to download and execute script asynchronously without waiting for it to execute and blocking the render process of the page . In order to increase build performance of our application we can target to improve the network performance.

3.8.8. Caching Facility:

Using Next JS we can set the cache control header of our application by setting header property in file named next.config.js .Content delivery network (CDN) is a network of server distributed world wide where we can store static assets in the server which gets delivered to user based on their IP region Next JS also provides built in support of Vercel Edge Network as a CDN automatically.

3.8.9. Bundle Analysis and Tree Shaking Techniques:

Bundle analysing is an approach for analysing our application to identify opportunity for decreasing it's size. Some tools for analysing the size of our application includes next-bundle analyser , webpack-bundle analyser etc. Tree shaking is the process of removing non essential, unused and dead code from the deployable bundle, we can use ES6 modules instead of common JS to take advantage of the Next JS builtin support for tree shaking . We can use import to import statement instead of requiring a module export statement. Next Js also helps us to exclude non essential folder or files from the bundling and build process , by this we can reduce the bundling time that improves overall performance by excluding specific non essential directories, we can use exclude parameter in the filename called next.config.js to exclude specific directory or file , by excluding non essential resource from deployment bundle we will decrease the amount bandwidth of data that needs to be processed , which leads to faster build times and efficient resource utilisation.

3.8.10. Server Side Rendering :

The Next JS framework already support server side rendering in rendering webpage ,so it will not make the user to see blank page in the initial load and reduced the load on browser. Next.js will load the HTML first then Javascript which makes the site visible , then internally it works to make the site interactive by loading the JavaScript. Next JS supports pre-rendering with server side rendering and server site generation method, so it can be adjusted based on application needs . In server site generation mechanism , pre rendering method generates the HTML during the build process.

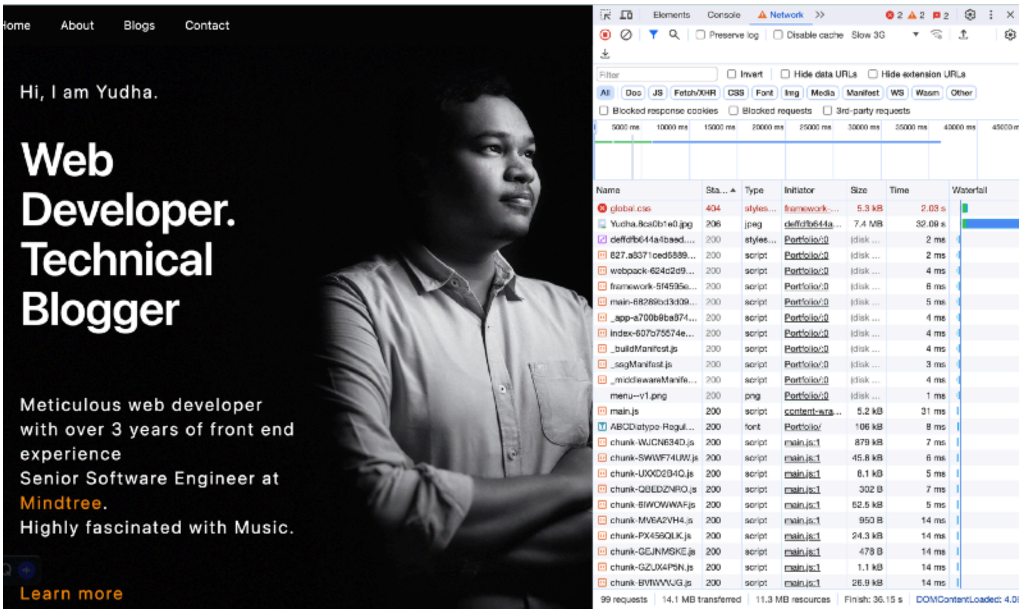


Figure 12: Next JS loading styles before Javascript

3.8.11. Easy Routing Feature:

Next js framework is having a easy routing mechanism or support located in the page folder. So it doesn't need a third party library to do the routing .

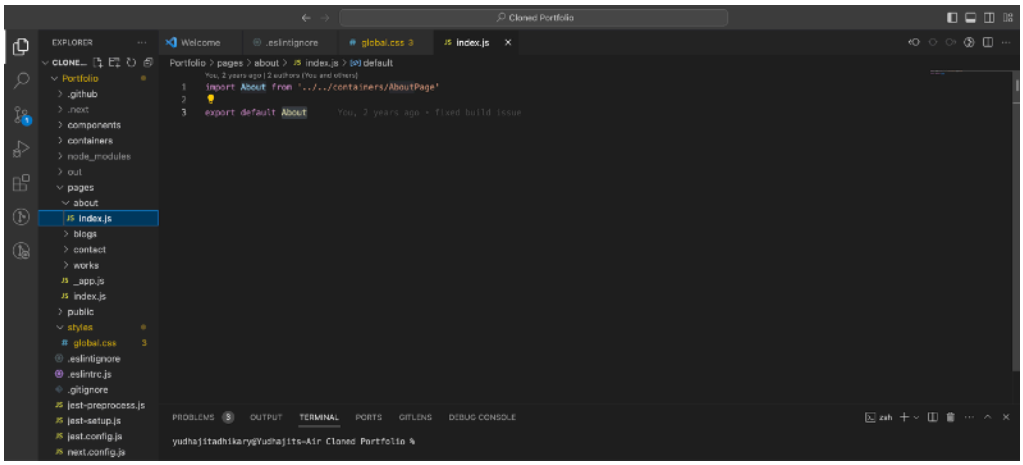


Figure 13: Page folder for routing mechanism in Next JS

3.8.12. SEO Benefits:

Next JS provides various data fetching mechanism , because it supports data fetching mechanism using client side rendering , server side rendering , server site generation and incremental static regeneration. Next JS has automatic code splitting , a technique of separating the application bundle into smaller chunk, this technic are known as code-splitting to reduce the initial load time of the application. Next JS supports dynamic imports , some component are imported during the run time and other components will get imported whenever it's necessary, this concept is known as lazy loading. This deferred landing helps to improve the performance of the website by decreasing the amount of Javascript.Next JS uses the server side rendering , which converts the react code to html on the initial request. A single page application transmit a large Javascript to the browser , majority of the HTML will be generated by the Javascript. When SEO google bots crawls the website, examining the HTML of each url they came across, the SEO bots do not composite Javascript and they do not always want it to be converted into HTML ,since bot determines what appears on the 1st page of search result it sometimes unable to read the page , So Google will not suggest or consider it as a best option to answer to web search .Next JS helps us the encapsulate the single page application which retains the SEO benefits of the traditional website architecture .

3.8.13. Key Optimisation Features of Next JS:

According to Dinku, Zerihun (2022) Next js incorporates developer experience with production ready features like , server side rendering , typescript capability , smart bundling , route prefetching ,optimised format of image , lazy loading , dynamic routing , file based routing and dynamic import etc. According to Desamsetti, Harshith ,Dekkati, Sreekanth(2021) Next Js helps to optimize the website performance by focusing on the server components, simplifying the process of obtaining data, making applications using new page directory, online streaming and internal tooling. According to Fariz M, Lazuardy S, Anggraini D (2022) Next JS already has built-in css support , in Next JS boilerplate we are already having a built-in css support , At the beginning of the installation there is a

global.css file and modules , this global.css is a css file in the Next Js framework that is globally accessible throughout the application.

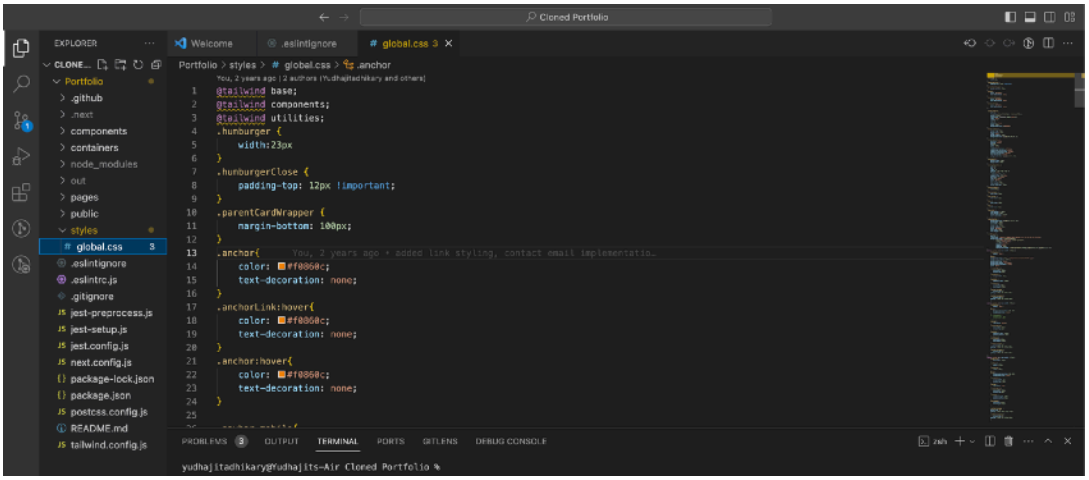


Figure 14: Built-in global css file in Next JS

Now let's summarise the key optimising features of Next JS in tabular format .

Optimisation	Description
Incremental Static Rendering	Next JS provides incremental static rendering concept which reduce the user delays without overwhelming the site with multiple resources , incremental static rendering generates HTML file and used to refresh or regenerate the HTML file again after certain amount of time rather than only once like static site generator for in every request like server side rendering .
Reduction of Javascript Bundle Size	Next JS reduces the Javascript bundle size on initial load by using code splitting and react lazy load and suspense component which loads necessary components or modules when needs or when user come to that exact fold, which leads to faster page load time and improved user experience.
Internalisation support with next-i18next	Next JS allows us to develop language and country specific pages which provides ability to change or switch between them using dropdown having language menu , this improves the user experience of the entire website.

Optimisation	Description
Optimisation of image with next optimize images	Next JS allows the optimisation of image of the site , rendering the overall page load time on improving the user experience , as well as improving the SEO by reducing the size of the page .

Table 1: Key optimisation features of Next JS

3.8.14. Best Practices for using Next JS:

Next.js is versatile and can benefit various industries, including e-commerce, healthcare, finance, education, and entertainment. Its scalability, performance, and flexibility make it suitable for a wide range of applications across different sectors. Let 's see the best practices we should follow to create a robust , scalable , flexible and optimized website using Next JS.

Best Practice	Description
Use NextJS for SSR	Next JS allows server side rendering which reduces the initial loading time by allowing content updates without delay, which can improve the performance and Seo score of the website .
Implement code splitting using React Lazy load	Next JS can reduce the size of JavaScript bundle by using lazy loading and suspense feature which improves the performance and user experience of the website .
Optimising image using Next image optimisation	Next JS is having image optimisation feature which helps to optimise the size of the website , reducing the page load time and improves the user experience and SEO.
Implement international support using next-i18next	Next JS allows us to develop language and country specific pages which provides ability to change or switch between them using dropdown having language menu , this improves the user experience of the entire website.
Use incremental static regeneration of content update	Next JS allows content updates to be reflected on the site without overwhelming resource which reduces initial load time and improves the user experience.

Table 2: Best Practices for developing Optimised application using Next JS

3.8.15. Benefits of Next JS:

Next JS provides solution for developing an performance optimised website . A optimised , seamless , well performing website will give a good user experience and will always ensure good SEO score of the site . According to Patel, Vishal (2023) site having good SEO score will be recommended by google first whenever user searches for any related keyword or queries which will increase the number of organic traffics of the site. Let's summarise what business impact Next JS can make by building a performance optimised website .

Benefit	Description
Improved performance and SEO	Implementation of Next JS comes with several optimisation techniques which significantly improve the website performance or SEO which includes faster page load time.
Enhanced user experience	Optimisation techniques such as internalisation support , reduced JS bundle size improves the user experience by improving page load time which gives smother , more seamless browser experience.
Increased organic traffic and revenue	Website having improved SEO and performance helps to increase the organic traffic as it provides good experience so users are more likely to visit or engage with the website.

Table 3: Benefits of developing Optimised application using Next JS

3.9. Real World Application

In this paper , we will be building a real world application to evaluate the performance impact of Next JS .We will be developing a static developer portfolio site using React JS and Next JS. The application will have four following pages .

3.9.1. HomePage:

It contains the short introduction of the developer.

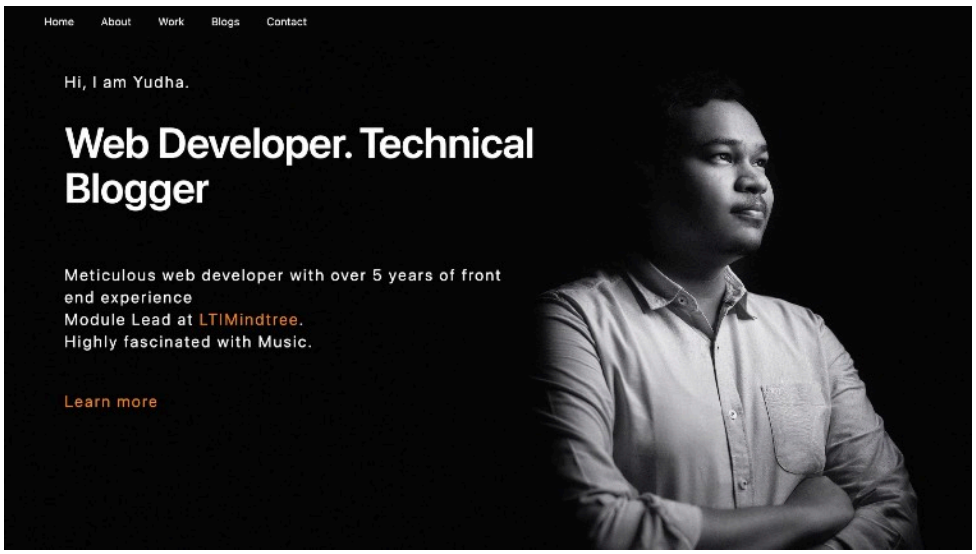


Figure 15: Homepage of Application

3.9.2. AboutPage:

It will have the detailed description of tech stack of the developer.

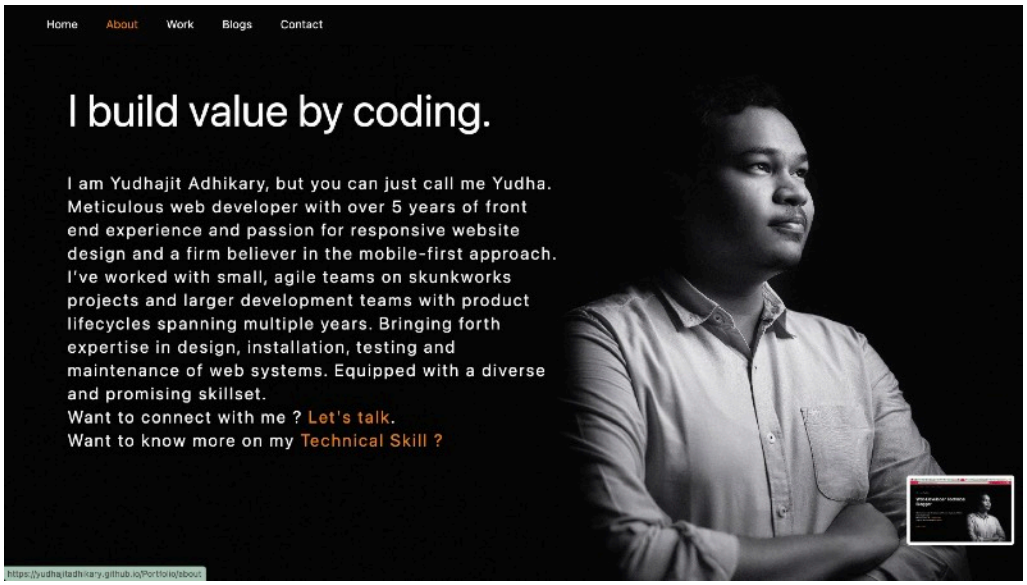


Figure 16: AboutPage of Application

3.9.3. BlogPage:

It will have the list of all technical blogs written by the developer , the blogs are linked to actual blog post so that user can navigate and read the blogs.

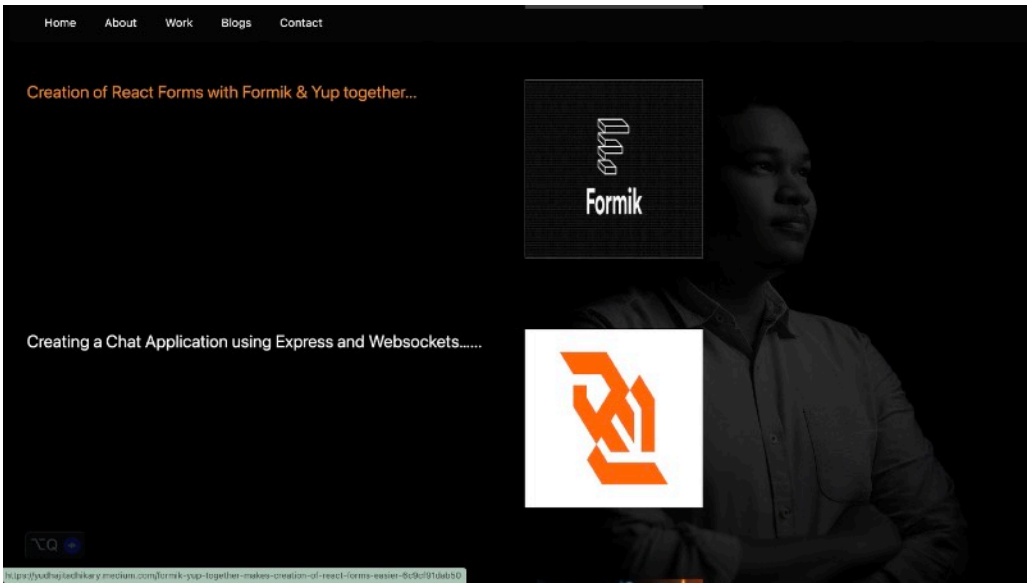


Figure 17: BlogPage of Application

3.9.4. ContactPage:

It will have developer's contact details, and anyone can directly reach out to the developer via email by filling the form provided in the contact page.Whenever a user fills the form a mail will be triggered to developer with all the queries and details filled in the form .

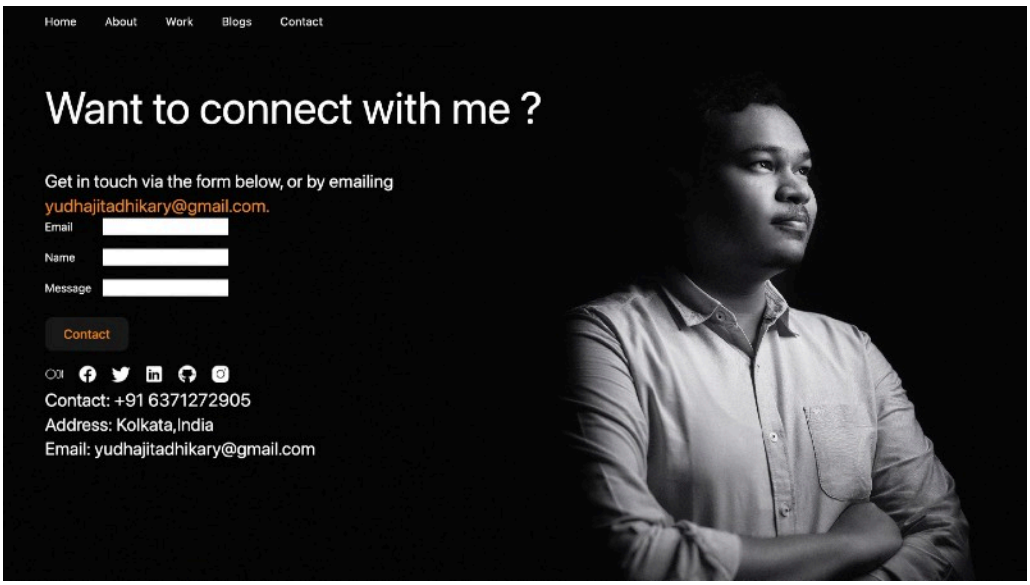


Figure 18: ContactPage of Application

3.10 Building with React JS :

React JS is having it's own boilerplate which makes easier for developer to start with coding. So we will first run the command **npx create-react-app <project name>** , this will create the boilerplate of react application, which looks like this.

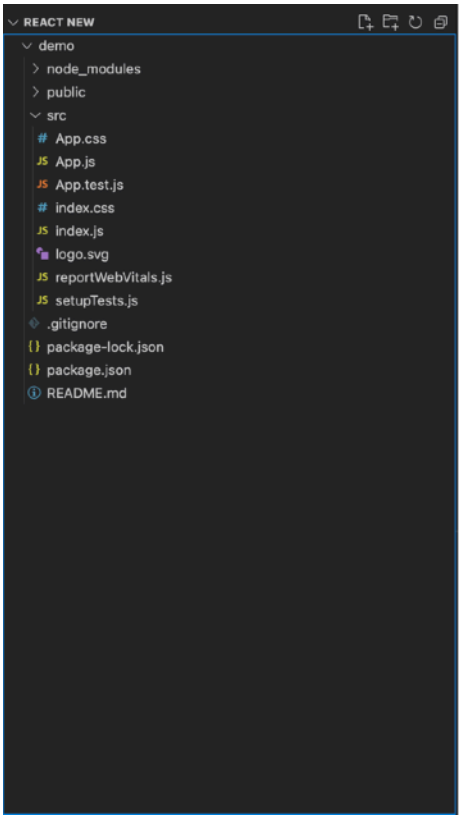


Figure 19: Folder Structure of React Application

So the boilerplate codebase is having **package.json** which consist of all the dependency npm packages we are using on our application, When we run **npm i** to install the packages mentioned in the **package.json** , 2 files will get generated , **package-lock.json** and **node_modules**, **package-lock.json** describes the exact npm dependency tree that is generated , **node_modules** acts as a cache for the external modules on which our project is dependent , it stores the npm package which are downloaded from web, **public** folder will store all the static contents and assets we are using in our application.

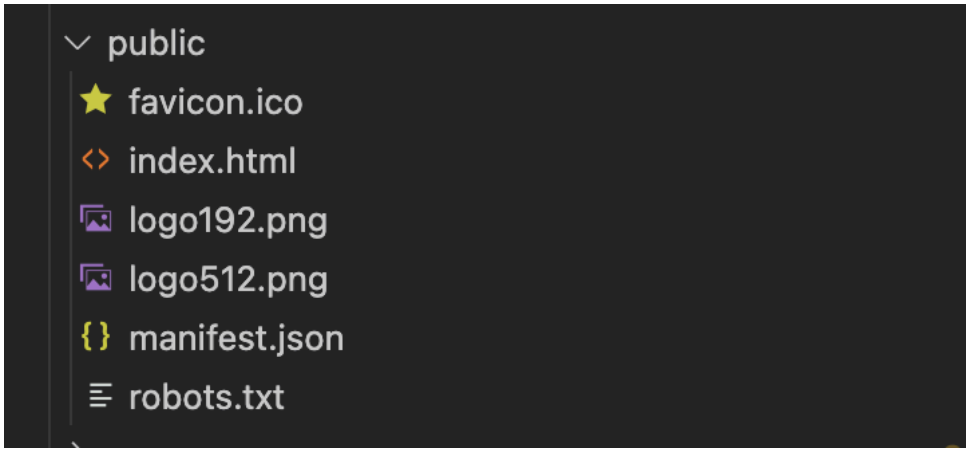


Figure 20: Public folder of React Application

README.md explains the details of the project , src folder contains the actual code of our application. Our code starts from index.js where we are importing **App.js** , it contains the actual HTML code that we will be seeing when we run the code in our local by running the command **npm start** .

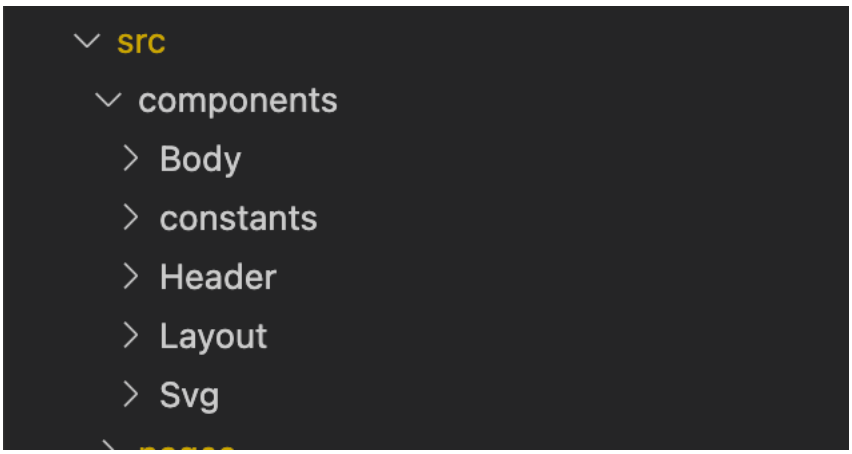


Figure 21: Components folder of React Application

We have added components folder inside **src** which contains all the components of our application .

We have also added **pages** folder which will contain all the pages of our application.

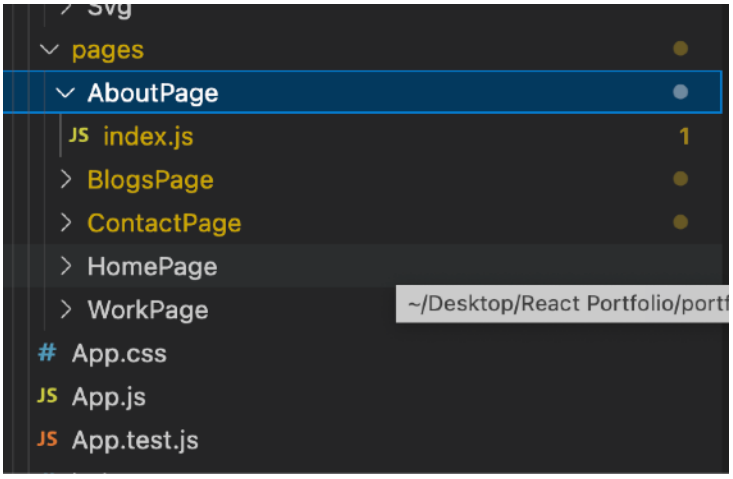
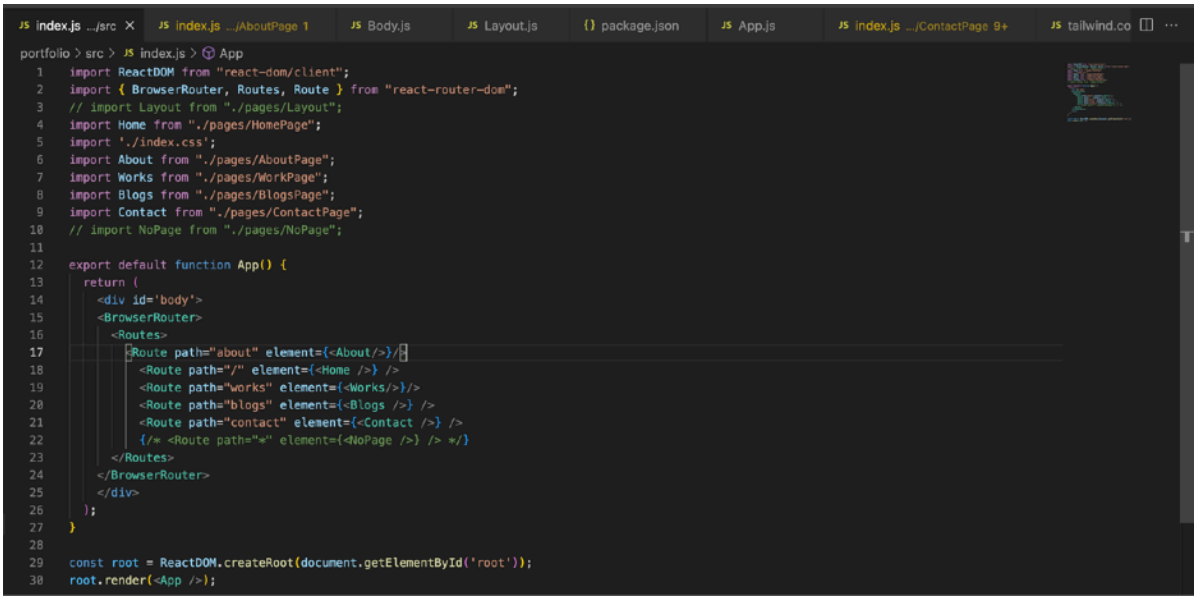


Figure 22: Pages folder of React Application

Before going to **index.js**, we should also consider that react does not support routing by itself , so we have to use npm package called **react-router-dom** to handle routing .In index.js we are importing all the pages of our application.Then we are defining our App component where we are mapping those pages with the respective routes using **react-router-dom**. Finally we are creating a dom having id “root” using React DOM and inside that DOM we are rendering our App component .



```
import ReactDOM from "react-dom/client";
import { BrowserRouter, Routes, Route } from "react-router-dom";
// import Layout from "./pages/Layout";
import Home from "./pages/HomePage";
import './index.css';
import About from "./pages/AboutPage";
import Works from "./pages/WorkPage";
import Blogs from "./pages/BlogsPage";
import Contact from "./pages/ContactPage";
// import NoPage from "./pages/NoPage";

export default function App() {
  return (
    <div id='body'>
    <BrowserRouter>
      <Routes>
        <Route path="about" element={<About/>}/>
        <Route path="/" element={<Home />} />
        <Route path="works" element={<Works/>}/>
        <Route path="blogs" element={<Blogs />} />
        <Route path="contact" element={<Contact />} />
        {/* <Route path="*" element={<NoPage />} /> */}
      </Routes>
    </BrowserRouter>
    </div>
  );
}

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(<App />);
```

Figure 23: index.js of React Application

We are using tailwind css for styling , so we have added **tailwind.config.js** file to define some basic styling related configuration for tailwind .

Figure 24: tailwind.config.js file of React Application

3.11 Building with Next JS

Now we will be developing the same application using Next JS , Next JS is build on top of React JS only , so like React JS , Next JS is also having a boilerplate available.So in order to get Next JS boilerplate installed in our system we will run **npx create-next- app@< version of Next>**. The Next JS boilerplate looks like this:

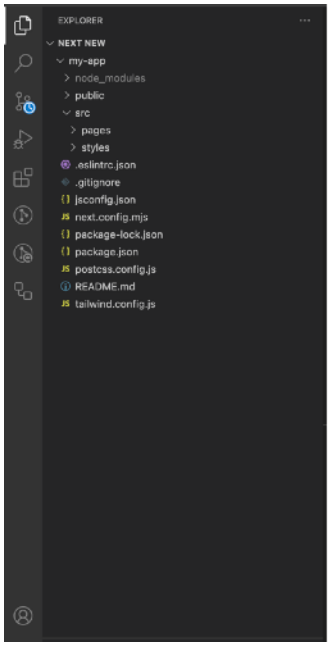


Figure 25: Basic boilerplate of Next JS

Next JS boilerplate is almost similar to React JS boilerplate , it is more segregated and atomic than react boilerplate, Next JS boilerplate is having **tailwind.config.js** by default, **postcss.config.js** is a JS file that transform our css code into an abstract syntax tree . It supports linting , minifying and injecting vendor prefixes. It produces fast build time compared to other processor . All the custom configuration of postcss can be added on **postcss.config.js** file.

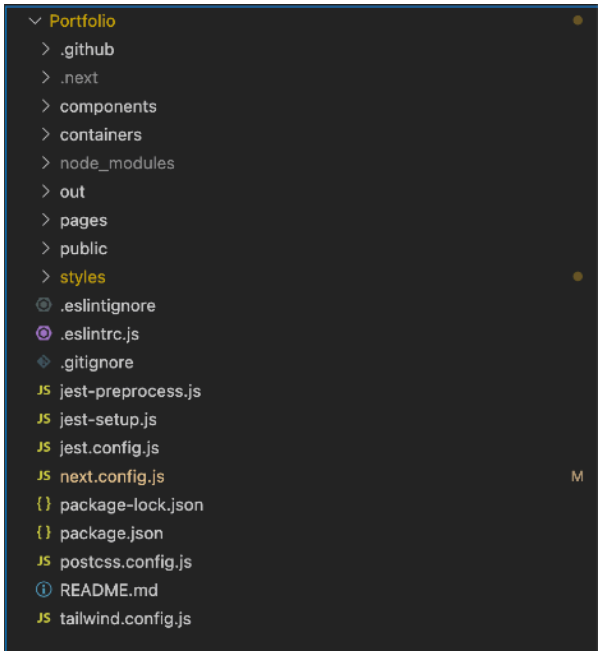


Figure 26: Folder structure of Next application

Next.config.js is a file which is used to configure Next JS configuration of our application. So to utilise the optimising feature of Next JS we will be using next/image instead of normal <img> tag , next/image is the extension of HTML <img> element with feature of automatic image optimisation, it also serves correctly sized image for each device, prevent from layout shift, it also helps in lazy loading the images , by loading images only when user enters to the viewport, So to use an image in next/image we have to whitelist the domain of the image for that we have to pass the image hostname or domain in the domains array on nextjs.config.js .

Figure 27: next.config.js of Next application

style folder contains all the css files , **global.css** is the default css file for Next JS , public folder will contain all the static assets of the application.

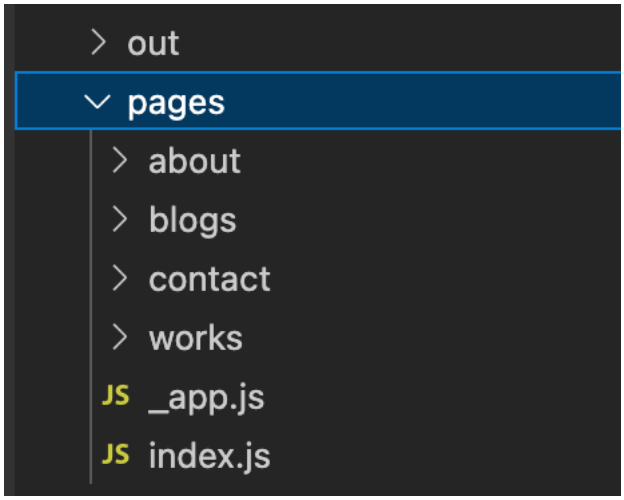


Figure 28: pages folder of Next application

Next JS is having pages folder by default , pages folder will have 3 files index.js , **_app.js** and **document.js** , **document.js** acts as the header of the application , where we can add all the third party script and tags required for our application , we can utilise next/script instead of normal script tag for adding the tags and script for our application . Next JS supports file based dynamic routing , for example in our case the code present in index.js inside pages folder will render on the base route of our application and will be considered as homepage of our application , similarly we have added the about pages as well inside pages folder , where code present in about/index.js will render on /about route and will be considered as AboutPage of our application .

```
import '../styles/global.css'
import React from 'react'
import dynamic from 'next/dynamic'
import PropTypes from 'prop-types'
const Layout=dynamic(()=>import('../components/Layout/Layout'))
You, 2 months ago | 2 authors (Yudhajitadhikary and others)
function MyApp({ Component, pageProps }) {

    return( <>
    <head>
    {/* <link rel='stylesheet' type='text/css' href='../styles/global.css'/> */}
    <link rel="preload" as="image" href="Yudha.avif"/>
    </head>
        <Layout>
            <div id='body'>
            <Component {...pageProps}/>
            </div>
        </Layout>
    </>)
}
MyApp.propTypes={
    Component:PropTypes.object,
    pageProps:PropTypes.object
}

export default MyApp

```

Figure 29: _app.js of Next application

Our code starts from **_app.js** , **_app.js** acts as the root directory of our application. We will be wrapping our application with Layout component , Layout component will be common components of our application across the page which will consist of header component which is common across the site , We will be passing the page content of our application as the children of Layout component .

```
Portfolio > components > Layout > JS Layout.js > ...
You, 2 months ago | 1 author (You)
import React from 'react'
import dynamic from 'next/dynamic'
import PropTypes from 'prop-types'
const Header=dynamic(()=>import('../Header/Header'))

const Layout=({children})=>{
    return(
        <React.Fragment>
            <Header/>
            <main>
                {children}
            </main>
            {/* <Footer/> */}
        </React.Fragment>
    )
}
Layout.propTypes={
    children:PropTypes.any.isRequired
}
export default Layout
```

Figure 30: Layout component of Next application

Now in Layout.js we are importing header dynamically using **next/dynamic** , it helps to import the component only when user scroll to the viewport , we are importing all the components using next/dynamic throughout the application, if the component is imported using next/dynamic the component will not be included in the page's initial JavaScript

bundle. The page will render the Suspense first, followed by the component when the boundary is resolved. Instead of using anchor tag we are using next/link. Next/link is extend of HTML <a> tag to provide prefetching and client side navigation between routes. **Next/link** use to prefetch when a <link> component enters within the viewport. Next JS prefetch and load the linked routes and data in background to improve the performance of the website. We are also using next/image instead of normal image tag where we can pass explicit height , width , quality and priority as well , like if the image is in first fold we can keep priority as true and if the image is in last fold we can keep priority as false .

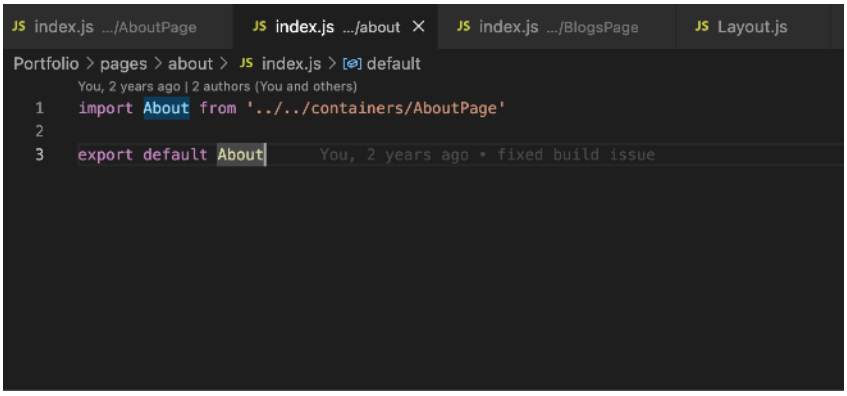


Figure 31: AboutPage of Next application

So our codebase is having pages folder , for example if we consider about/index.js file inside pages folder , we are importing AboutPage from container which is having the actual code of AboutPage .

Figure 32: Index.js of AboutPage of Next application

Container folder will have the actual code of the page. Once the coding is done we can run the code in our local. Before running the local we have to run **npm run build** to create deployable chunks of our application , the deployable chunks will be stored in **out** folder and **.next** folder will be created to store page cache and some feature to speed up our application. Once build is done we can run npm run start to run the application in our local.

3.12. Deploying with Github pages

Now we will be hosting both the react version and next version of the application using Github pages . Github pages allows us to host a webpage from our repository. First we have to signup into Github. The Github will be creating a domain in this format <Github Username> followed by .github.io, something like **<Github Username>.github.io** .So first we signed up on Github , created two empty repository with name react-portfolio for React version and Portfolio for Next version of our application. On those repository we will push the entire code of our application. In order to publish our webpage on the Github pages we installed npm package called **gh-pages**, and we added two scripts on the package.json of our application .

“predeploy": "react-scripts build” (for React application)

"deploy": "gh-pages -d build” (for React application)

“predeploy”: “next build” (for Next application)

“deploy”: “gh-pages -d build” (for Next application)

We will be deploying our application by running **npm run deploy** script. It will first run **predeploy** script which will create deployable chunks of the application then it will be running **gh-pages -d build** which will deploy the code using gh-pages.

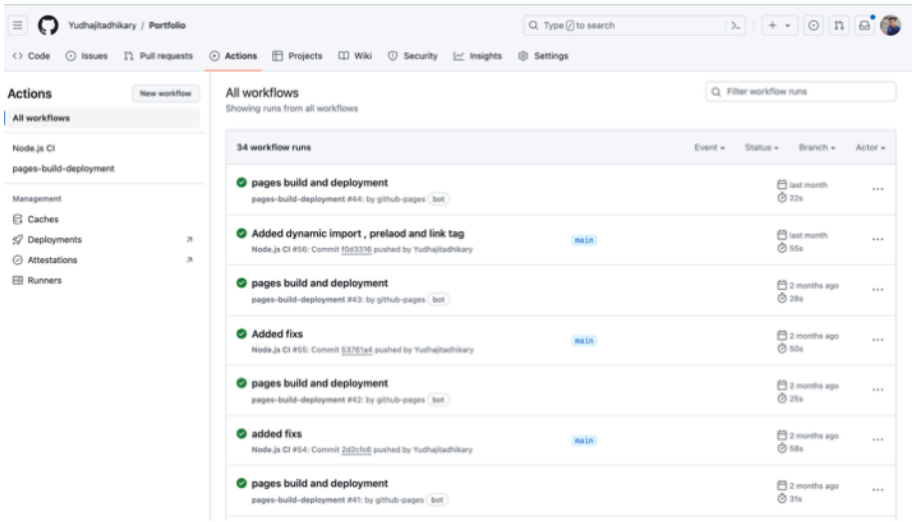


Figure 33: All build pipeline of Github pages

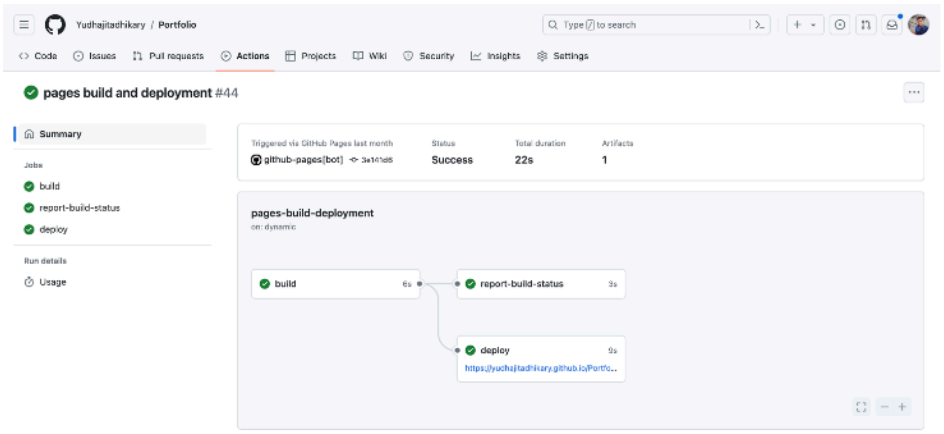


Figure 34: Build and deployment Slot of Github pages

So in this way we have deployed React JS version and Next JS version of our application on **https://yudhajitadhikary.github.io/react-portfolio/** and **https// yudhajitadhikary.github.io/Portfolio** domain respectively .

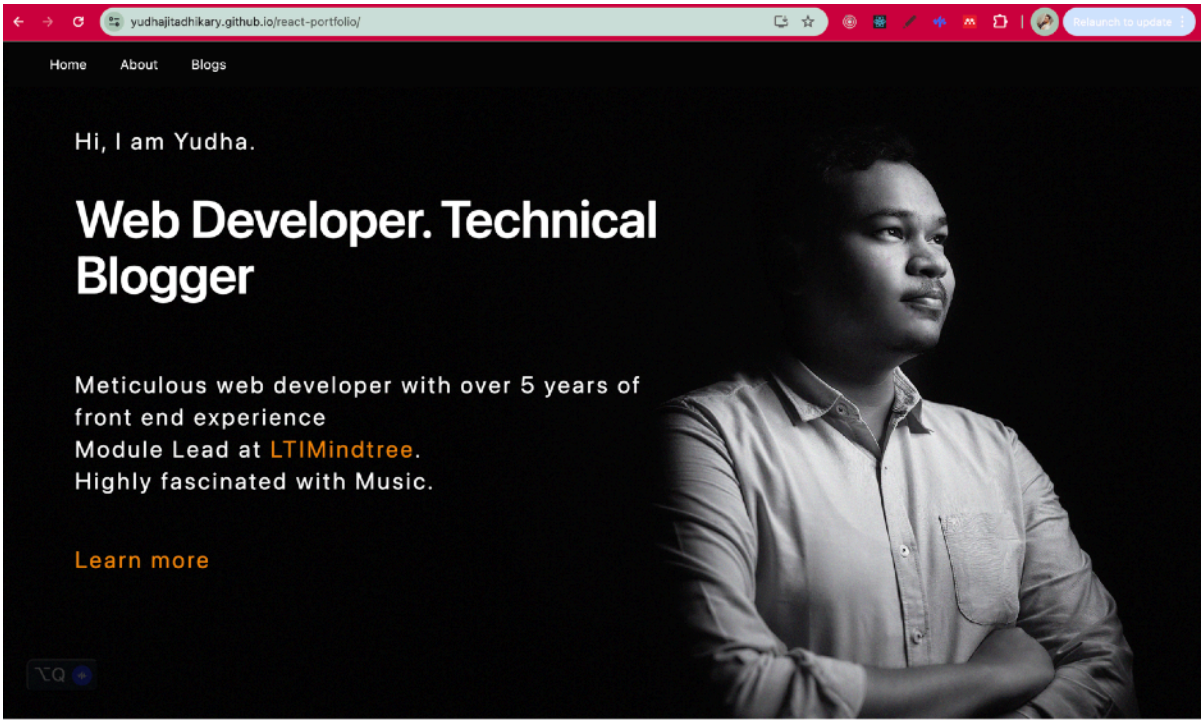


Figure 35: React JS Application

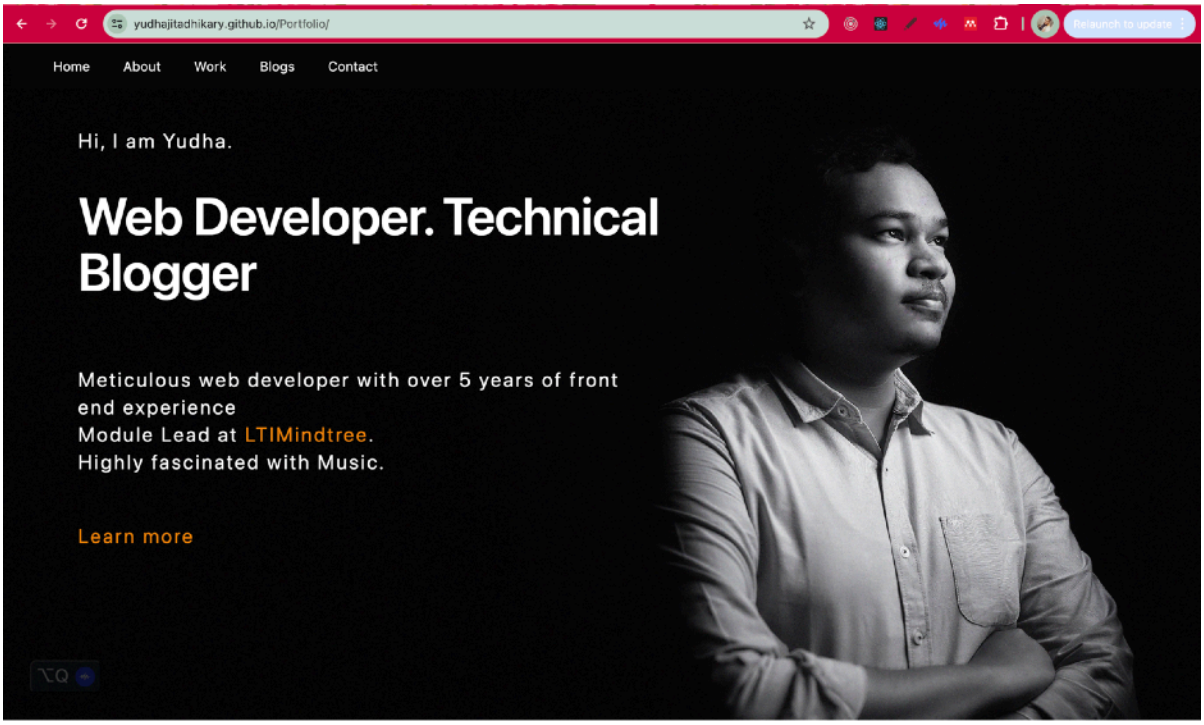


Figure 36: Next JS Application

3.13. Data Analysis Procedure:

Now we have both version of the application build and deployed on domains, To evaluate the impact of Next JS on page speed improvement, we will be running page speed scan using google page speed insights. We will be performing these activity five times to get a consistent and accurate result on page performance. The page speed insights reports on the user experience of the page on both mobile and desktop device. It measures the real user experience data and is powered by the chrome user experience report (CrUX). Pagespeed Insights reports core real user experience metrics like First Contentful Paint (FCP) , Interaction on Next Paint (INP) , Cumulative Layout Shift (CLS) and Largest Contentful Paint(LCP), it also reports for experimental metrics Time to First Byte (TTFB) as well as the deprecated metrics First Input delay (FID).

3.13.1 Core web vitals Parameters:

Google using their ongoing continuous engagement and collaboration experience with millions of developer and site owners have developed many helpful opportunities to improve user experience , web vitals is an initiative by Google to provide unified guidance for providing good experience to the website . The user experience quality have many variants , some aspects are user experience specific and some are content specific . So core web vitals will be the measuring parameter of website in this paper , Core web vitals brings all the aspect of website under one umbrella whether it's visual stability , initial loading experience, effectiveness in user interaction etc.

According the google page speed insight there are 3 metrics which can be considered and efficiently used to affect the page speed of our application those are Largest Contentful Paint (LCP) ,Cumulative Layout Shift (CLS) and Interaction on Next Paint (INP) , which replaced First Input Delay (FID) very recently .

3.13.1.1. Largest Contentful Paint (LCP)

Largest Contentful Paint is a metric which measures when a page to page navigation appears completed to a user , ideally 2.5 second for 75% of their page should load.

Figure 37: LCP Threshold

3.13.1.2. Cumulative Layout Shift (CLS)

Most of our application need to load lots of elements so browser use to load these element progressively so user can start making progress to achieve their intension without waiting for other things , but if a layout shift happens the position of already loaded , visible component will shift to other position , which will have negative impact on the overall user experience and user can commit several errors as well due to the shift . For example if user want to buy a product by clicking buy now button and due to layout shift cancel button comes on the portion of buy now button they can mistakenly click on cancel order button , this is really a worst scenario which will lead to cancellation of the order which user want to order and that will negatively affect the user experience . CLS is the metric which exactly measures the layout shift issue of our application.

Figure 38: CLS Threshold

3.13.1.3. First Input Delay (FID)

According to Kaaresoja, T.,Brewster, S., & Lantz, V. (2014) visual delay from touching button become perceived when it ranges from 70 ms to 100ms , but people rate button as slow when it gets increased from 100 ms to 150 ms . Aside from changing how the UI feels, delaying something people expect to be near-instantaneous can lead them to make errors. They may repeat an action because they think it didn't work, and the second action can have an undesirable effect. First Input Delay is a metrics which exactly measure this delay of response of the website.

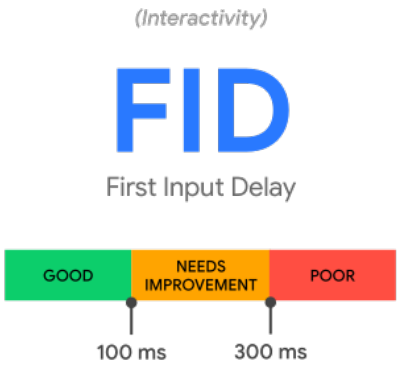


Figure 39: FID Threshold

3.13.1.4. Interaction to Next Paint (INP)

From March 2024, INP is introduced as the successor metric to First Input Delay (FID). While both are responsiveness metrics, FID only measured the input delay of the first interaction on a page. INP improves on FID by observing all interactions on a page.
It observes the latency of all click, tap , keyboard interaction within a page throughout the lifespan. Good responsiveness means a page respond quickly to interaction, when a page response to an interaction the browser present visual feedback in the next frame to show that the interaction is successful.

Figure 40: INP Threshold

	Good	Poor	Percentile
Largest Contentful Paint	<=2500ms	>4000ms	75
First Input Delay	<=100ms	>300ms	75
Cumulative Layout Shift	<=0.1	>0.25	75
Interaction on Next Paint	<=200ms	>500ms	75

Table 5: Thresholds of core web vitals which categorise performance of website

3.13.2. Pagespeed Insights Standards:

PageSpeed classifies the quality of user experience into three buckets, Good , Needs Improvement and Poor.

The distribution is divided into the categories Good , Needs Improvements and Poor , which is represented by Green , Amber and Red bars .

The metrics scores and the perf score are coloured according to these ranges:

a> 0 to 49 (red): Poor

b> 50 to 89 (orange): Needs Improvement

c> 90 to 100 (green): Good

To provide a good user experience, sites should strive to have a good score (90-100). A "perfect" score of 100 is extremely challenging to achieve and not expected. For example , taking a score from 99 to 100 needs about the same amount of metric improvement that would take a 90 to 94. The Core Web vitals metrics are INP , LCP and CLS and they may be aggregated at either the page and origin level. If aggregation of these 3 metrics is 75th of percentiles , it will be marked as Good, if aggregation has insufficient data from INP, then it

will pass the assessment if both the 75th parameters of LCP and CLS are Good. We may get variation on the page speed score , several common sources are which are responsible for these variation are local network availability , client hardware availability and content resource contention.

Metrics	Good	Needs Improvement	Poor
FCP	[0,1800 ms]	(1800 ms, 3000 ms]	Over 3000 ms
LCP	[0,2500 ms]	(2500 ms, 4000 ms]	Over 4000 ms
CLS	[0,0.1]	(0.1, 0.25]	Over 0.25
INP	[0,200 ms]	(200 ms, 500 ms]	Over 500 ms
TTFB (Experimental)	[0, 800 ms]	(800 ms, 1800 ms]	Over 1800 ms

Table 4 : Core Web Vitals Threshold

So we are having 3 pages in our application , Homepage , Aboutpage and Blogpage .We will be conducting page speed scan on 5 consecutive dates from 7th Sept 2024 - 11th Sept 2024 using page speed insights on both React and Next JS version of the application, we will be calculating the average of all the 5 page speed scan performed in each days to get the average of each core web vitals parameters, we will be collecting value of the core web vitals parameters namely : LCP,INP,CLS,TBT,FCP for both the desktop and mobile version, and we will be analysing the issues highlighted for each page in Diagnostics section .

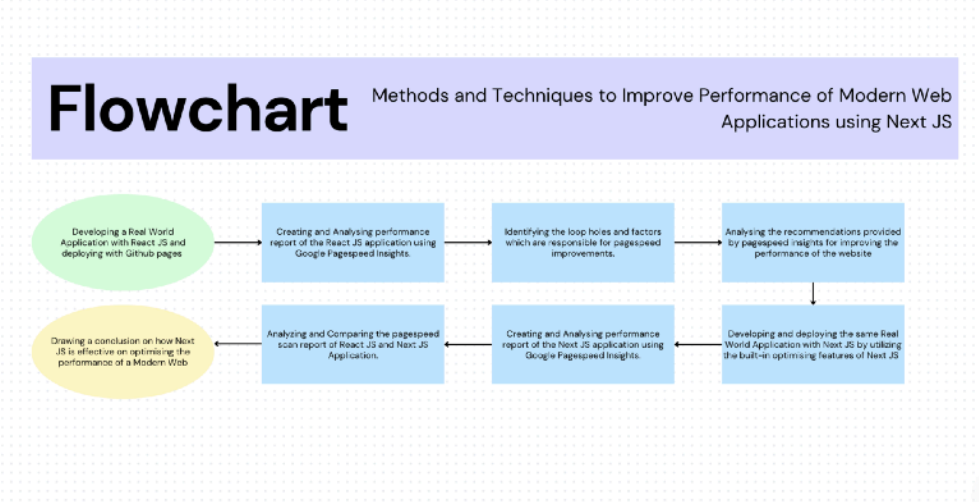


Figure 41: Flowchart of Research Methodology

3.14 Summary

In this research paper, we will be analysing how Next JS can improve the page-speed of a website. For that first we will be analysing what are the factors that affects the page-speed of a website . The main performance indicator chosen in this thesis project are the core web vitals metrics defined by Google . Web performance will be evaluated by the overall page speed score and score we are getting for each of the core web vitals metrics .Then we will explore and analyse how Next JS can leverage the page-speed of a website. After that we will be conducting page-speed scan on a reference static site using Google Page-speed insights/LightHouse , which is a tool for improving the quality of web pages and is able to run a variety of test against a webpage while monitoring various performance like the core web vitals and speed index.Test will be conducted 5 times on Google Chrome browser using MacBook Air M1 2020. The important metrics we will consider while comparing and measuring the website performance are :

- First Contentful Paint (FCP): measures the time required from starting page loading to rendering any part of the page .
- Largest Contentful Paint (LCP): measures the time required to start loading the page to rendering the largest text block on the screen.
- Interaction to Next Paint (INP): measures the latency of every action in the site like tap, keyboard interaction , click on the page . It measures the response time of the every action in the website.
- Total Blocking Time (TBT): measures how long the main thread is blocked which prevents user interaction and website responsiveness .
- Cumulative Layout Shift (CLS): measures the unexpected layout shift or sudden shifting of the components in the site .
- Time to First Byte (TTFB): measures the time taken for the network to respond to the first byte of the resource .

Then after analysing the scan report we will be developing the same application using Next JS and will conduct another page-speed scan on the application which is developed using Next JS . Finally we will compare and evaluate both the page speed scan report to draw a conclusion on how Next JS is impacting the page speed of the application .

3.15 Required tools

We will be using Google Page speed Insights for generating the website page speed report which we will be analysing.We will be using Node Js as the JavaScript runtime environment ,Visual Studio code as the code editor , Git as the distributed version control system , Github as software development platform for storing , tracking and collaboration of software development, Github pages as static site hosting service to deploy application, Mendeley for managing citation of our research paper . We will be using MacBook Air M1 2020 having Apple M1 chip with Mac OS , MacOS Sonoma 14.0 version for implementing all the tasks for this research paper.

4. Research Plan:

We will be following the below mentioned timelines to complete this research paper within the deadlines . Please find the below Gantt chart for the activities scheduled for completing the research paper.

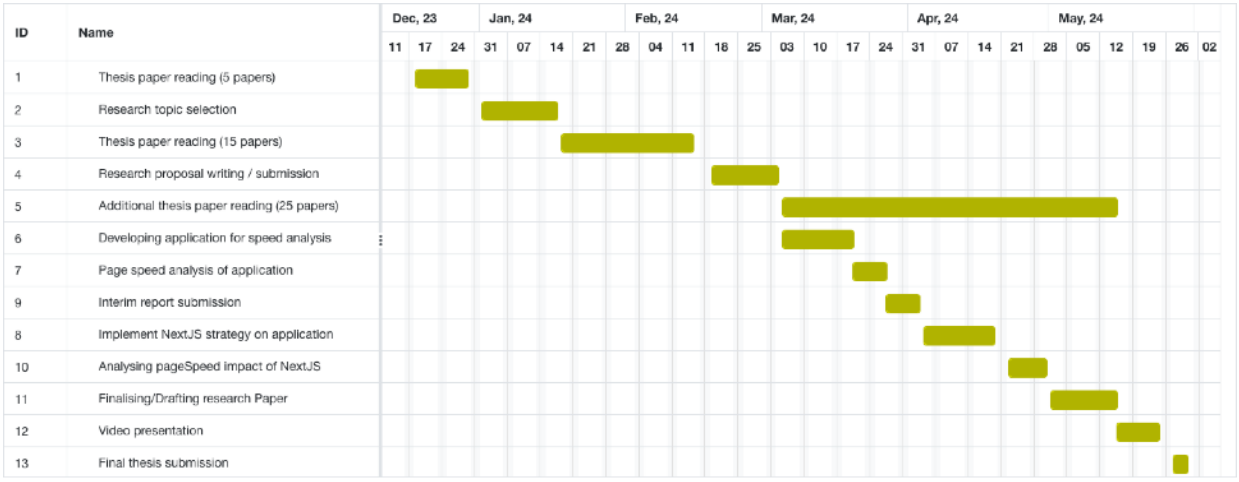


Figure 9: Gantt chart of Research Plan

APPENDIX B: Website Details:

React JS Website Url: **https://yudhajitadhikary.github.io/react-portfolio/**

React JS Github repo link : **https://github.com/Yudhajitadhikary/react-portfolio**

Next JS Website Url: **https://yudhajitadhikary.github.io/Portfolio/**

Next JS Github repo link: **https://github.com/Yudhajitadhikary/Portfolio**