

IMPLEMENTASI PROGRESSIVE WEB APPS PADA WEBSITE GETHELP MENGUNAKAN NEXT.JS

Oleh:

Ronny Jubhari Phie Joarno¹, Mohammad Fajar^{2*}, Arfan Yunus³

^{1,2,3}Sistem Informasi, STMIK Kharisma Makassar

e-mail: ¹ronnyjubhari_19@kharisma.ac.id, ²fajar@kharisma.ac.id,

³arfanyunus@kharisma.ac.id

Abstrak: Penelitian ini bertujuan untuk mengimplementasikan teknologi Progressive Web Apps (PWA) pada website GetHelp menggunakan framework Next.js sehingga website dapat diakses melalui home screen smartphone seperti aplikasi android dan dapat dijalankan secara offline. Teknik pengumpulan data dilakukan melalui studi literatur dan observasi performansi sistem melalui pengujian menggunakan PWA tools. Teknologi web yang digunakan dalam PWA terdiri dari web app manifest, app shell dan service worker. Implementasi yang dilakukan meliputi tahapan instalasi PWA Next.js, membuat web app manifest, membuat file registrasi service worker, mengimplementasikan service worker, menambahkan script untuk memanggil manifest.json dan app.js dan mengevaluasi PWA. Hasil evaluasi menunjukkan layanan-layanan website GetHelp dapat diakses dengan baik melalui beragam tipe perangkat mobile pada platform Android. Pengujian kualitas hasil implementasi PWA berdasarkan baseline PWA checklist dengan tool Lighthouse memperlihatkan semua kriteria Progressive Web Apps terpenuhi, sedangkan pada tool PWABuilder memberikan skor total 110/110. Hal ini dapat disimpulkan bahwa implementasi PWA pada website GetHelp telah memenuhi kriteria Progressive Web Apps. Selain itu, hasil evaluasi performa sistem menggunakan tools Lighthouse didapat peningkatan sebesar 23% (sebelum implementasi 66%, setelah implementasi PWA, performa menjadi 89%). Sementara jika evaluasi menggunakan tool GTMetrix, performa website GetHelp naik 14% (sebelum implementasi sebesar 68%, setelah implementasi PWA menjadi 82%).

Kata kunci: Progressive Web Apps, PWA, Next.js, GetHelp, Donasi Online

Abstract: The aim of this study is to implement Progressive Web Apps (PWA) on the GetHelp website using Next.js so that the website can be accessed via the smartphone home screen like an android application and can be run offline. Data collection techniques are carried out through literature studies and system performance observations through testing using PWA tools. The web technologies used in PWAs consist of the web app manifest, app shell and service worker. The implementation includes the steps of installing PWA Next.js, creating a web app manifest, creating a service worker registration file, implementing a service worker, adding a script to call manifest.json and app.js and evaluate the PWA. The evaluation results show that GetHelp website services can be accessed well through various types of mobile device on the Android platform. The results of quality testing of PWA implementation based on the baseline PWA checklist with the Lighthouse tool shows that all Progressive Web Apps criteria are met, while the PWABuilder tool gives a total score of 110/110. It can be concluded that the implementation of PWA on the GetHelp website has met the criteria for Progressive Web Apps. In addition, the results of evaluation system performance using Lighthouse tools increase by 23% (before implementation 66%, after PWA implementation, performance was 89%). Meanwhile, if the evaluation uses the GTMetrix tool, the performance of the GetHelp website increases by 14% (before implementation by 68%, after implementation of PWA to 82%).

Keywords: Progressive Web Apps, PWA, Next.js, GetHelp, Online Donation

* Corresponding author : Mohammad Fajar (fajar@kharisma.ac.id)

1. PENDAHULUAN

GetHelp yang saat ini dapat diakses di alamat <https://gethelpid.com> merupakan salah satu platform donasi ataupun galang dana secara *online* yang bertujuan untuk menyediakan media bagi masyarakat atau para donatur terkait donasi dan pengelolaannya sehingga penggalangan dana secara *online* dapat dilakukan secara transparan dan akuntabel. Meskipun layanan yang tersedia pada GetHelp telah dapat diakses melalui aplikasi browser pada perangkat PC ataupun laptop, akan tetapi tidak demikian halnya bagi pengguna yang sehari-harinya menggunakan perangkat mobile. Bagi kelompok pengguna perangkat mobil tersebut akan menemukan kesulitan dalam berinteraksi dengan sistem Gethelp. Hal ini disebabkan desain awal layanan Gethelp tidak mempertimbangkan pengakses melalui platform mobile, sementara pada umumnya kebanyakan masyarakat menggunakan perangkat mobile untuk mengakses informasi atau layanan di internet. Terlebih lagi sejumlah kompetitor yang juga menyediakan layanan penggalangan dana *online* telah mendukung platform mobile untuk mengakses layanan-layanannya. Di sisi lain mengembangkan aplikasi berbasis mobile secara khusus akan membutuhkan biaya baik dari aspek waktu maupun tenaga mengingat perangkat mobile memiliki beragam jenis termasuk keragaman platform sistem operasinya. Oleh karena itu, studi ini bertujuan untuk mengimplementasikan Progressive Web Apps (PWA) pada website GetHelp menggunakan *framework* Next.js sehingga layanan website dapat diakses melalui home screen smartphone seperti aplikasi android dan dapat dijalankan secara *offline*.

Teknologi PWA dikembangkan oleh Google bekerja sama dengan pengembang browser yang bertujuan untuk membuat aplikasi multi-platform menjadi semakin mudah dan cepat [1]. Dimana pembuatan website melibatkan teknologi seperti service worker, web manifest, dan cache API [2]. PWA memiliki kelebihan dimana user mampu mengakses website tanpa perlu menggunakan browser lagi karena seperti aplikasi Android, lebih cepat, bekerja *offline* dan mudah diakses melalui home screen [3]. Selain itu, PWA juga dapat mengubah aplikasi web biasa yang hanya dapat diakses melalui web browser menjadi sebuah aplikasi hybrid [4], yang bersifat cross platform atau dapat di install di beragam platform sistem operasi seperti windows, android dan iOS [5]. User juga tidak perlu khawatir harus menginstal aplikasi yang menggunakan banyak ruang pada memori perangkat. Maka dari itu PWA memudahkan para pengguna sistem agar dapat mengakses aplikasi lebih cepat melalui perangkat apapun seperti smartphone ataupun PC [6]. Next.js merupakan kerangka kerja fleksibel yang dapat digunakan untuk membuat aplikasi web dengan cepat. Untuk membuat antarmuka pengguna (User Interface) yang interaktif, Next.js memerlukan react sebagai *library* Javascript. Implementasi PWA pada website GetHelp menggunakan *framework* Next.js dengan pertimbangan beberapa hal, seperti proses rendering website dilakukan di sisi server (*server side rendering*) sehingga halaman website lebih cepat ditampilkan di browser, lebih *seo friendly*, performa website lebih baik, setup dan *deploy project* mudah dilakukan, memungkinkan developer untuk membuat aplikasi web dengan menerapkan server side

rendering dan melakukan generate static website dengan mudah tanpa harus melakukan konfigurasi apapun (zero config).

Implementasi Progressive Web Apps telah dikaji dalam literatur. Seperti dalam penelitian [7] yang dilakukan Nurwanto berjudul Penerapan Progressive Web Application (PWA) pada E-Commerce menggunakan CMS (Content Management System) Wordpress dengan theme Metro Store dan plugins Woocommerce untuk membuat e-commerce. Penelitian [8] yang dilakukan Gharizi Matiini, Rahmat Setiyadi, Adi Setiawan, dan M. Ramli berjudul Pengembangan Aplikasi Progressive Web Application (PWA) Untuk Pembelajaran Dan Evaluasi Kelas English Grammar Online Course. Sedangkan, pada penelitian yang dilakukan Dedi Haryanto dan Zulhipni Reno Saputra Elsi [9] yang berjudul Analisis Performance Progressive Web Apps Pada Aplikasi Shopee. Pada rancangan objek penelitian menggunakan GTMetrix. Selain itu, pada penelitian yang dilakukan Aiyatul Mu'in, Amri dan Anwar [10] berjudul Monitoring Sistem Keluhan Mahasiswa Menggunakan Progressive Web Application Pada Politeknik Negeri Lhokseumawe, dibangun sebuah aplikasi monitoring sistem keluhan mahasiswa menggunakan PWA di jurusan Teknologi Informasi dan Komputer menggunakan *framework* Laravel. Demikian pula, penelitian yang dilakukan Grace Levina Dewi, Suhatati Tjandra, dan Ricardo [11] berjudul Pemanfaatan Progressive Web Apps Pada Web Akuntansi, Progressive Web Apps diimplementasikan pada sistem informasi akuntansi untuk meningkatkan kinerja serta reliabilitas data menggunakan *framework* Laravel dan *Quasar*. Dalam studi ini, digunakan *framework* yang berbeda dengan studi-studi sebelumnya, yaitu Next.js. Selain itu juga diukur tingkat terpenuhinya kriteria sebuah PWA dan performa website sebelum dan sesudah implementasi PWA.

2. METODE PENELITIAN

2.1. Jenis Data dan Sumber Data

Dalam penelitian ini, data primer diperoleh dari *tools* yang digunakan untuk pengujian yaitu Lighthouse, PWABuilder dan GTMetrix, sementara untuk data sekunder didapatkan dari studi literatur berupa artikel jurnal yang berkaitan dengan penelitian yang menyebutkan salah satu solusi efektif bagaimana layanan website bisa diakses seperti aplikasi *mobile* atau *cross platform* tanpa membangun ulang kode program untuk setiap platform yaitu menggunakan teknologi Progressive Web Apps (PWA).

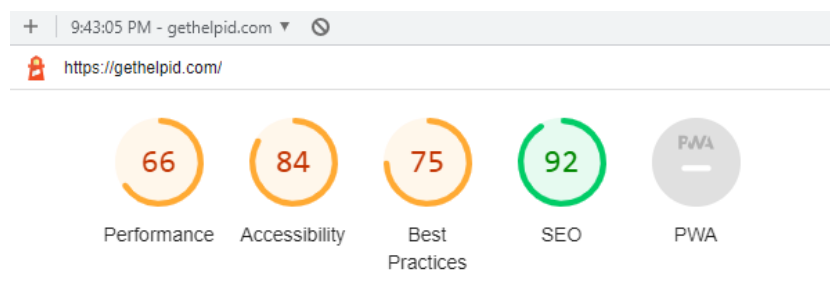
Setelah memahami penerapan PWA maka dilakukan observasi terhadap website GetHelp untuk mengumpulkan data website sebelum dan setelah dilakukannya implementasi. Observasi dilakukan terhadap website GetHelp untuk melihat kualitas website baik dari segi performa website maupun terpenuhinya kriteria sebuah PWA dengan menggunakan *tools* Lighthouse, PWABuilder, dan GTMetrix yang diuji ke website GetHelp. Pada penelitian ini diambil data pengujian kriteria PWA dan data performa website. Untuk pengujian tingkat terpenuhinya kriteria sebuah PWA berdasarkan baseline PWA checklist atau persyaratan dasar PWA menggunakan Lighthouse yang terdiri dari 2 parameter (installable, PWA

Optimized) dan pengujian manual dan PWABuilder terdiri dari 3 parameter dengan masing-masing 11 kriteria yang diuji. Pada pengujian menggunakan Lighthouse, setiap kriteria yang diuji ingin diketahui apakah PWA yang diimplementasikan sudah memenuhi kriteria sebuah PWA atau tidak, sedangkan pengujian menggunakan PWABuilder, setiap kriteria yang diuji diberikan nilai 10 dengan skor tertinggi yaitu 110, sedangkan pengujian performa website menggunakan tools GTMetrix terhadap 12 variabel dan Lighthouse dengan 7 variabel pengujian. Data kriteria PWA dan performansi website yang dikumpulkan sebelum implementasi PWA disajikan pada Tabel 1 hasil evaluasi kriteria PWA dari Lighthouse dan PWABuilder, Gambar 1 hasil evaluasi performansi menggunakan Lighthouse, dan Gambar 2 hasil evaluasi performansi dari GTMetrix.

Tabel 1. Data Kriteria PWA Pada Lighthouse dan PWABuilder

Tools	Kriteria Terpenuhi
Lighthouse	Ukuran konten tepat untuk viewport
	Memiliki tag <meta name="viewport"> dengan atribut width atau initial-scale
PWABuilder	Menggunakan HTTPS
	Memiliki sertifikat SSL yang valid
	Tidak ada konten campuran di halaman

Hasil evaluasi awal terhadap kriteria PWA sebelum PWA diimplementasikan didapatkan 2 kriteria terpenuhi menggunakan Lighthouse dan 3 kriteria terpenuhi dari GTMetrix sedangkan kriteria lainnya belum terpenuhi. Hasil ini menunjukkan PWA belum diimplementasikan dan tidak memenuhi setiap kriteria PWA.



Gambar 1. Data Performa Website Sebelum Implementasi PWA Menggunakan Lighthouse

Performa website sebelum implementasi PWA menggunakan Lighthouse didapatkan performance sebesar 66%. Hasil evaluasi awal yaitu website memiliki performa yang masih rendah (tidak baik).



Gambar 2. Data Performa Website Sebelum Implementasi PWA Menggunakan GTMetrix

Pada tools GTMetrix didapatkan performa website yang masih rendah (tidak baik) dengan performance sebesar 68%.

Keberhasilan penerapan PWA ditentukan dari terpenuhinya kriteria yang dijadikan tolak ukur [12] yang disajikan pada Tabel 2 di bawah ini :

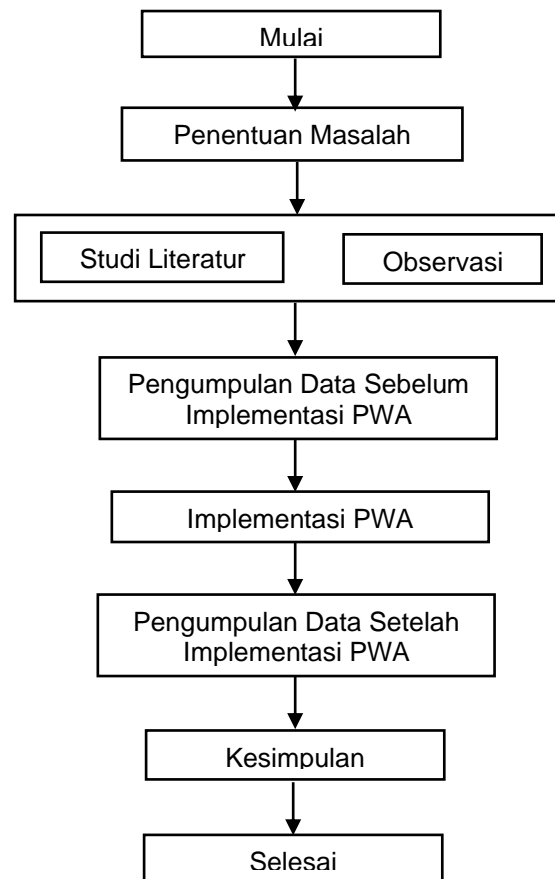
Tabel 2. Kriteria Keberhasilan Penerapan PWA

No.	Variabel
1.	Web app manifest atau service worker memenuhi persyaratan instalasi aplikasi
2.	Menggunakan HTTPS
3.	Mendaftarkan <i>service worker</i> yang mengontrol halaman dan start_url
4.	Konfigurasi untuk splash screen kustom
5.	Mengatur warna tema untuk address bar
6.	Ukuran konten tepat untuk viewport
7.	Mempunyai tag <meta name="viewport"> dengan atribut width atau initial-scale
8.	Menyediakan apple-touch-icon yang valid
9.	Manifest memiliki maskable icon
10.	Situs dapat dijalankan di berbagai browser (cross-browser)
11.	Transisi halaman terasa cepat dan tidak terasa seperti diblokir di jaringan
12.	Setiap halaman memiliki URL

Masing-masing kriteria memiliki persyaratan yang harus dipenuhi agar suatu website dapat dikatakan benar-benar berhasil mengimplementasikan PWA.

2.2. Tahapan Penelitian

Ada beberapa tahapan implementasi PWA yang dilakukan penulis yaitu pada tahap awal dimulai dengan menginstall PWA Next.js (next-pwa) kemudian membuat file web app manifest dengan nama manifest.json. Setelah itu membuat file registrasi service worker dengan nama app.js untuk mendaftarkan service worker pada sistem. Selanjutnya membuat Service Worker Next.js. Setiap file konfigurasi PWA yang telah dibuat pada localhost komputer kemudian di upload ke server web hosting DomaiNesia. Pada tahapan terakhir, dilakukan penambahan script untuk memanggil manifest.json pada bagian header dan app.js pada bagian footer website. Untuk tahapan penelitian disajikan pada Gambar.



Gambar 3. Tahapan Penelitian

3. HASIL DAN PEMBAHASAN

3.1. Implementasi PWA

Pada tahap ini, penulis melakukan pengimplementasian PWA yang terdiri dari implementasi teknologi web app manifest, app shell, dan service worker. Langkah awal yaitu menerapkan *framework* Next.js pada website gethelp. *Framework* ini telah menyediakan teknologi pendukung dalam membangun PWA seperti service worker. Pertama dilakukan instalasi PWA Next.js dengan sebuah baris perintah seperti pada Gambar 4.

```
> npm install next-pwa
```

Gambar 4. Perintah Install PWA Next.js

Setelah itu membuat web app manifest dengan nama manifest.json. Manifest.json digunakan untuk menyimpan konfigurasi aplikasi PWA diantaranya yaitu nama aplikasi, deskripsi, ikon, background color, theme color dan orientation. Selain itu, dengan manifest.json dapat memunculkan pop-up "tambahkan ke layar utama" pada browser dan menampilkan splash screen ketika aplikasi web dibuka. File manifest.json dapat dilihat pada Gambar 5 di bawah ini.

```
1 {
2   "name": "GetHelp",
3   "short_name": "GetHelp",
4   "description": "Platform Donasi & Galang Dana",
5   "theme_color": "#ffffff",
6   "background_color": "#ffffff",
7   "display": "standalone",
8   "scope": "/",
9   "start_url": "/",
10  "icons": [
11    {
12      "src": "assets/img/icon-36x36.png",
13      "sizes": "36x36",
14      "type": "image/png"
15    },
16    {
17      "src": "assets/img/icon-96x96.png",
18      "sizes": "96x96",
19      "type": "image/png"
20    },
21    {
22      "src": "assets/img/icon-192x192.png",
23      "sizes": "192x192",
24      "type": "image/png"
25    },
26    {
27      "src": "assets/img/pwa-icon-maskable-512x512.png",
28      "sizes": "512x512",
29      "type": "image/png",
30      "purpose": "maskable"
31    }
32  ],
33  "splash_pages": null
34 }
```

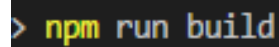
Gambar 5. File manifest.json

Tahap berikutnya yaitu membuat file registrasi service worker dengan nama app.js untuk mendaftarkan service worker pada sistem. Service worker adalah script yang berjalan di background website dan dengan adanya service worker, website dapat diakses secara *offline* [13]. Kode file registrasi service worker ditunjukkan pada Gambar 6 di bawah ini.

```
1 if ('serviceWorker' in navigator) {
2   window.addEventListener('load', function () {
3     navigator.serviceWorker.register('/sw.js').then(function(registration){
4       console.log('Service Worker registered');
5     }, function(err) {
6       console.log('Service Worker registration failed');
7     });
8   });
9 }
```

Gambar 6. Konfigurasi Registrasi Service Worker Next.js

Selanjutnya mengimplementasikan service worker yang telah disediakan *framework* Next.js melalui terminal Visual Studio Code. Dengan baris perintah dibawah ini, service worker otomatis dibuat seperti ditunjukkan pada Gambar 7.



```
> npm run build
```

Gambar 7. Pembuatan Service Worker Next.js

Service worker adalah tool yang sangat kuat pada PWA yang menyediakan fungsionalitas *offline*, push notification, pembaruan konten, caching konten, dan banyak lagi lainnya [14]. Service worker memiliki kemampuan untuk menyimpan aset-aset sebuah website seperti file HTML, CSS, Javascript, dan gambar di penyimpanan browser, sehingga ketika website tersebut diakses tanpa koneksi internet service worker mengembalikan aset website yang sudah disimpan sebelumnya, hal ini membuat sebuah website dapat bekerja seperti aplikasi native Android dan desktop [15]. File service worker yang telah dibuat otomatis oleh *framework* Next.js lebih memudahkan dalam implementasi PWA karena sudah tidak perlu membuat sendiri.

Setiap file konfigurasi PWA yang telah dibuat pada localhost komputer kemudian di upload ke server web hosting DomaiNesia. Struktur folder server pada DomaiNesia sebelum implementasi PWA disajikan pada Gambar 8 di bawah ini.

Sebelum Implementasi		Setelah Implementasi	
Name	Size	Name	Size
system	4 KB	.gitignore	517 bytes
user_guide	4 KB	.htaccess	1.39 KB
.editorconfig	302 bytes	app.js	528 bytes
.gitignore	517 bytes	composer.json	62 bytes
.htaccess	1.39 KB	composer.lock	46.11 KB
composer.json	62 bytes	contributing.md	6.68 KB
composer.lock	46.11 KB	GETHELPBACKEND.zip	32.97 MB
contributing.md	6.68 KB	index.php	9.98 KB
GETHELPBACKEND.zip	32.97 MB	license.txt	1.09 KB
index.php	9.98 KB	manifest.json	1.66 KB
license.txt	1.09 KB	page.html	0 bytes
page.html	0 bytes	readme.rst	2.29 KB
readme.rst	2.29 KB	robots.txt	100 bytes
robots.txt	100 bytes	sitemap.xml	5.08 KB
sitemap.xml	5.08 KB	sw.js	1.23 KB

Gambar 8. Struktur folder server pada DomaiNesia

Tahap terakhir yaitu menambahkan script untuk memanggil manifest.json pada bagian header website disajikan pada Gambar 9.

```
<link rel="stylesheet" href="<?= base_url('assets/css/responsive.min.css') ?>">
<link rel="shortcut icon" href="<?= base_url('assets/img/profile/') ?>default.jpeg">
<link rel="manifest" href="<?= base_url('manifest.json') ?>">
```

Gambar 9. Pemanggilan manifest.json dan app.js pada Header Website

Pemanggilan app.js pada bagian footer halaman website disajikan pada Gambar 10.

```
<script src="<?= base_url('app.js') ?>"></script>
```

Gambar 10. Pemanggilan app.js pada Footer Website

3.2. Pembahasan

Progressive Web Apps yang telah diimplementasikan kemudian diuji pada berbagai perangkat untuk mengetahui apakah bisa dijalankan pada smartphone dan mampu diakses seperti aplikasi Android bahkan dalam kondisi jaringan tidak bagus atau *offline* sekalipun. Pengujian pada berbagai jenis perangkat dapat dilihat pada Tabel 3 di bawah ini.

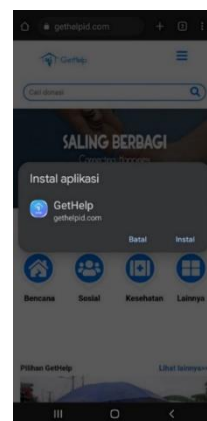
Tabel 3. Spesifikasi Perangkat Testing

Perangkat	OS	Layar	Internal Storage	RAM	Browser
Samsung Galaxy A10s	Android 11	6.2" (720 x 1520) pixels	32GB	2GB	Chrome
Realme 3	Android 10	6.2" (720 x 1520) pixels	64GB	3GB	Opera
Oppo A3s	Android 8.1	6.2" (720 x 1520) pixels	32GB	3GB	Firefox

Salah satu hasil implementasi PWA pada perangkat Samsung Galaxy A10s dapat dilihat pada Gambar 11. Jika pilihan tersebut diklik maka ditampilkan informasi permintaan install atau menambahkan ke layar utama perangkat seperti ditunjukkan pada Gambar 12 di bawah ini.



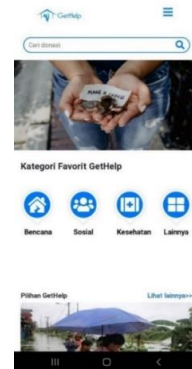
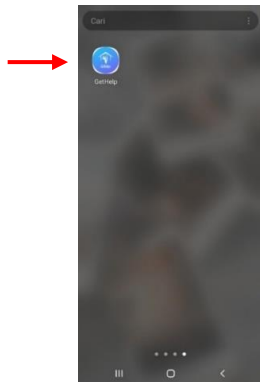
Gambar 11. Add to Homescreen



Gambar 12. Instal PWA ke Layar Utama

Pada Gambar 11 dapat dilihat pilihan tambahkan ke layar utama telah muncul artinya web app manifest telah bekerja dengan baik. Jika sudah menambahkan aplikasi web ke perangkat mobile maka muncul icon aplikasi di layar utama perangkat yang ditampilkan pada

Gambar 13. Saat user membuka PWA tersebut halaman utama website GetHelp seperti pada Gambar 14.



Gambar 13. Icon Aplikasi di Layar Utama

Gambar 14. Halaman Utama GetHelp

Pada Gambar 14, saat aplikasi PWA dibuka, tampilannya terlihat seperti aplikasi native biasa atau aplikasi android namun sebenarnya itu adalah sebuah website di mana web tersebut sudah berhasil diimplementasikan PWA. Pada tahap berikutnya dilakukan pengujian terhadap performa PWA dan kriteria PWA berdasarkan *baseline PWA checklist*.

Pengujian Baseline PWA Checklist

Pengujian kriteria PWA menggunakan Lighthouse didapatkan hasil seperti ditampilkan pada Tabel 4 di bawah ini.

Tabel 4. Hasil Pengujian Baseline PWA Checklist Menggunakan Lighthouse

No.	Variabel	Sebelum Implementasi	Setelah Implementasi
Installable			
1.	Web app manifest atau service worker memenuhi persyaratan instalasi aplikasi	Tidak Terpenuhi	Terpenuhi
PWA Optimized			
2.	Mendaftarkan <i>service worker</i> yang mengontrol halaman dan <i>start_url</i>	Tidak Terpenuhi	Terpenuhi
3.	Konfigurasi untuk splash screen kustom	Tidak Terpenuhi	Terpenuhi
4.	Mengatur warna tema untuk address bar	Tidak Terpenuhi	Terpenuhi
5.	Ukuran konten tepat untuk viewport	Terpenuhi	Terpenuhi
6.	Mempunyai tag <code><meta name="viewport"></code> dengan atribut <i>width</i> atau <i>initial-scale</i>	Terpenuhi	Terpenuhi
7.	Menyediakan <i>apple-touch-icon</i> yang valid	Tidak Terpenuhi	Terpenuhi
8.	Manifest memiliki <i>maskable icon</i>	Tidak Terpenuhi	Terpenuhi
Pengujian Manual			
9.	Situs dapat dijalankan di berbagai browser (<i>cross-browser</i>)	Tidak Terpenuhi	Terpenuhi
10.	Transisi halaman terasa cepat dan tidak terasa seperti diblokir di jaringan	Tidak Terpenuhi	Terpenuhi
11.	Setiap halaman memiliki URL	Tidak Terpenuhi	Terpenuhi

Tools Lighthouse memberikan hasil pengujian berupa warna yaitu **hijau** untuk kriteria yang **terpenuhi** dan **merah** untuk kriteria **tidak terpenuhi** [12]. Pengujian PWA pada website GetHelp berdasarkan *baseline PWA checklist* dilakukan dengan 3 cara yaitu menggunakan Lighthouse, PWABuilder dan pengujian manual. Pengujian dengan *tools* Lighthouse terdiri dari 2 parameter yaitu Installable dan PWA Optimized dengan total 8 kriteria, sedangkan pengujian manual terdiri dari 3 kriteria. Dari hasil pengujian *baseline PWA checklist* pada website GetHelp menggunakan Lighthouse, setelah pengimplementasian PWA didapatkan hasil dimana semua kriteria pengujian terpenuhi berbanding dengan sebelum implementasi PWA dimana 2 kriteria pengujian yang terpenuhi. Hasil ini menunjukkan Progressive Web Apps (PWA) telah berhasil diimplementasikan dan memenuhi semua kriteria sebuah PWA.

Pengujian Baseline PWA Checklist Menggunakan PWABuilder

Tools PWABuilder memberikan skor untuk menentukan setiap kriteria terpenuhi atau tidak, jika terpenuhi maka diberikan skor 10 dan jika tidak terpenuhi maka skornya 0 (nol). Pada pengujian PWA berdasarkan *baseline PWA checklist* menggunakan PWABuilder didapatkan hasil seperti ditampilkan pada Tabel 5 di bawah ini.

Tabel 5. Hasil Pengujian Baseline PWA Checklist Menggunakan PWABuilder

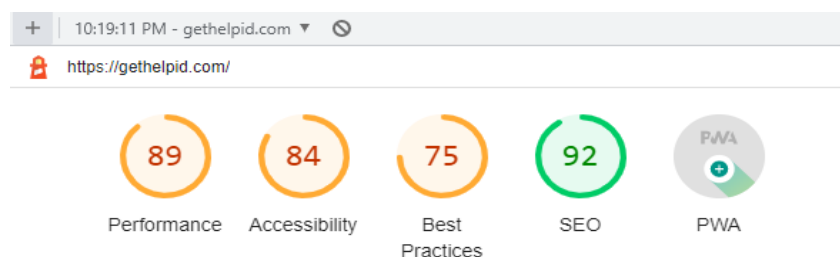
No.	Variabel	Sebelum Implementasi		Setelah Implementasi	
		Status	Nilai	Status	Nilai
1.	Web Manifest terpasang dengan benar	Tidak Terpenuhi	0	Terpenuhi	10
2.	Memiliki daftar icon untuk ditambahkan ke Layar Beranda	Tidak Terpenuhi	0	Terpenuhi	10
3.	Memiliki property name	Tidak Terpenuhi	0	Terpenuhi	10
4.	Memiliki property short_name	Tidak Terpenuhi	0	Terpenuhi	10
5.	Menentukan start_url	Tidak Terpenuhi	0	Terpenuhi	10
6.	Memiliki PNG icon 512x512 atau lebih besar	Tidak Terpenuhi	0	Terpenuhi	10
7.	512x512 atau ikon yang lebih besar berhasil dimuat dari jaringan	Tidak Terpenuhi	0	Terpenuhi	10
8.	Memiliki service worker	Tidak Terpenuhi	0	Terpenuhi	10
9.	Menggunakan HTTPS	Terpenuhi	10	Terpenuhi	10
10.	Memiliki sertifikat SSL yang valid	Terpenuhi	10	Terpenuhi	10
11.	Tidak ada konten campuran di halaman	Terpenuhi	10	Terpenuhi	10
Total			30		110

Pada pengujian menggunakan PWABuilder didapatkan perbandingan antara website sebelum implementasi PWA dan setelah implementasi PWA. Sebelum website diimplementasikan PWA, skor yang didapatkan sebesar 30 dimana 3 kriteria pengujian yang terpenuhi. Sedangkan setelah PWA diimplementasikan, didapatkan skor 110 dengan 11

kriteria pengujian yang terpenuhi pada sebuah PWA. Hasil pengujian pada PWABuilder menunjukkan PWA telah berhasil diimplementasikan dan memenuhi kriteria sebuah PWA.

Pengujian Performa Website Menggunakan Lighthouse

Pengujian performa website dilakukan dengan 2 cara yaitu menggunakan Lighthouse dan GTMetrix. Pengujian dengan *tools* Lighthouse terdiri dari 7 kriteria, sedangkan pengujian dengan GTMetrix terdiri dari 12 kriteria. Pengujian PWA menggunakan Lighthouse disajikan pada Gambar 15, sedangkan pengujian dengan GTMetrix disajikan pada Gambar 16.



Gambar 15. Performa Website Setelah Implementasi PWA

Berikut rincian aspek pengujian performa website menggunakan Lighthouse dapat dilihat pada Tabel 6 di bawah ini.

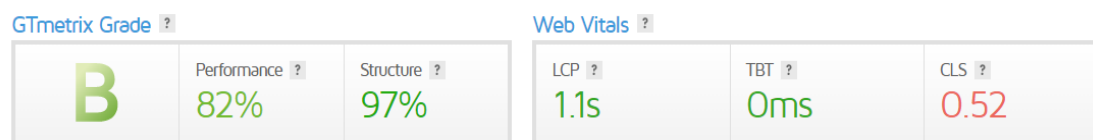
Tabel 6. Hasil Pengujian Performa Website Menggunakan Lighthouse

No.	Variabel	Sebelum Implementasi PWA	Setelah Implementasi PWA
1.	Performance	66%	89%
2.	First Contentful Paint	1.9s	1.9s
3.	Time to Interactive	5.3s	2.3s
4.	Speed Index	2.2s	2.0s
5.	Total Blocking Time	670ms	150ms
6.	Largest Contentful Paint	4.1s	3.3s
7.	Cumulative Layout Shift	0.001	0.001

Pada pengujian performa website menggunakan Lighthouse, setelah diimplementasikan PWA didapatkan hasil performa website yang lebih baik dibandingkan sebelum pengimplementasian PWA yang bisa dilihat pada Tabel 6 di atas. Performa website dari 66% menjadi 89% setelah implementasi PWA, durasi menampilkan konten (First Contentful Paint) memiliki hasil yang sama baik sebelum maupun setelah implementasi PWA, durasi halaman menjadi interaktif sepenuhnya (Time to Interactive) dari 5.3s menjadi 2.3s setelah implementasi PWA, pada Speed Index dari 2.2s menjadi 2.0s, durasi yang diblokir selama proses pemuatan halaman (Total Blocking Time) dari 670ms menjadi 150ms, durasi menampilkan konten terbesar (Largest Contentful Paint) dari 4.1s menjadi 3.3s, dan pada pengukuran pergerakan elemen yang terlihat di dalam area pandang (Cumulative Layout

Shift) memiliki hasil yang sama baik sebelum maupun setelah implementasi PWA. Hasil ini menunjukkan pengimplementasian PWA meningkatkan performa website.

Pengujian Performa Website Sebelum dan Setelah Implementasi PWA Menggunakan GTMetrix



Gambar 16. Performa Website pada GTMetrix Setelah Implementasi PWA

Berikut rincian aspek pengujian performa website menggunakan GTMetrix dapat dilihat pada Tabel 7 di bawah ini.

Tabel 7. Hasil Pengujian Performa Website Menggunakan GTMetrix

No.	Variabel	Sebelum Implementasi PWA	Setelah Implementasi PWA
1.	Performance	68% (Grade C)	82% (Grade B)
2.	First Contentful Paint	1.5s	1.1s
3.	Time to Interactive	1.5s	1.2s
4.	Speed Index	6.8s	1.3s
5.	Total Blocking Time	0ms	0ms
6.	Largest Contentful Paint	1.5s	1.1s
7.	Cumulative Layout Shift	0.52	0.52
8.	Redirect Duration	0ms	0ms
9.	Time to First Byte (TTFB)	758ms	755ms
10.	Fully Loaded Time	4.1s	3.8s
11.	Total Page Size	1.61 MB	1.49 MB
12.	Total Page Request	51	48

Pada pengujian performa website menggunakan GTMetrix, sebelum implementasi PWA didapatkan hasil performance 68% sedangkan setelah diimplementasikan didapatkan hasil performance yang lebih baik yaitu 82%, durasi menampilkan konten (First Contentful Paint) dari 1.5s menjadi 1.1s setelah implementasi PWA, durasi halaman menjadi interaktif sepenuhnya (Time to Interactive) dari 1.5s menjadi 1.2s setelah implementasi PWA, pada Speed Index dari 6.8s menjadi 1.3s, durasi yang diblokir selama proses pemuatan halaman (Total Blocking Time) memiliki hasil yang sama baik sebelum maupun setelah implementasi PWA, durasi menampilkan konten terbesar (Largest Contentful Paint) dari 1.5s menjadi 1.1s, dan pada pengukuran pergerakan elemen yang terlihat di dalam area pandang (Cumulative Layout Shift) dan durasi untuk mengarahkan ulang URL (Redirect Duration) memiliki hasil yang sama baik sebelum maupun setelah implementasi PWA, sedangkan Total waktu dari

request awal hingga saat menerima byte respons pertama (Time to First Byte) dari 758ms menjadi 755ms, pada durasi memuat satu halaman penuh website (Fully Loaded Time) dari 4.1s menjadi 3.8s, besar ukuran seluruh halaman website dari 1.61 MB menjadi 1.49 MB setelah implementasi PWA, dan durasi yang dibutuhkan website melakukan request (Total Page Request) dari 51 menjadi 48 setelah implementasi PWA. Hasil ini menunjukkan performa website meningkat setelah pengimplementasian PWA.

4. KESIMPULAN

Dari penelitian yang telah dilakukan oleh penulis, maka didapatkan kesimpulan sebagai berikut:

1. PWA yang diimplementasikan membuat website dapat diakses seperti aplikasi android pada perangkat mobile tanpa perlu menggunakan browser.
2. Implementasi PWA pada website Gethelp telah memenuhi semua kriteria PWA seperti dapat diakses melalui home screen smartphone, dapat dijalankan secara *offline* di berbagai browser dan meningkatkan performa website dibandingkan sebelum PWA digunakan.
3. Pengujian kualitas PWA berdasarkan *baseline PWA checklist* pada website GetHelp setelah implementasi PWA dengan Lighthouse mendapatkan hasil pengujian dimana semua kriteria PWA terpenuhi sedangkan dengan PWABuilder mendapatkan skor total 110/110. Dari 11 kriteria pengujian pada kedua tools, semuanya dapat diimplementasikan berarti sudah memenuhi kriteria PWA dan telah diimplementasikan dengan baik.
4. Adanya peningkatan performa website GetHelp sebesar 23% dari 66% sebelum implementasi PWA menjadi 89% setelah pengimplementasian PWA menggunakan Lighthouse dan pengujian menggunakan GTMetrix, performa website naik 14% dari 68% (Grade C) sebelum implementasi PWA menjadi 82% (Grade B) setelah implementasi PWA. Berbagai cara dapat dilakukan untuk meningkatkan performa website diantaranya dengan memperbaiki ukuran gambar yang digunakan pada website, mengurangi penggunaan css/javascript yang tidak digunakan dan yang mengganggu proses pemuatan konten, minify atau menghilangkan karakter yang tidak perlu pada css/js dan sebagainya.

DAFTAR PUSTAKA

- [1] Soleha, E. Budiman, and M. Wati, "Pengembangan Progressive Web Application Portal Program Studi Teknik Informatika Berbasis Restful API," *Pros. Semin. Nas. Ilmu Komput. dan Teknol. Inf.*, vol. 4, no. 2, pp. 115–120, Sep. 2019.
- [2] L. Adi, R. J. Akbar, and W. N. Khotimah, "Platform E-Learning untuk Pembelajaran Pemrograman Web Menggunakan Konsep Progressive Web Apps," *Tek. ITS*, vol. 6,

- no. 2, pp. A579–A583, 2017.
- [3] Chris Love, *Progressive Web Application Development by Example: Develop Fast, Reliable, and Engaging User Experiences for the Web*. Britania Raya: Packt Publishing, 2018.
- [4] E. S. Fadly and J. Budiarto, “Aplikasi Evaluasi Perkembangan Latihan Atlet Panahan Menggunakan Progressive Web Application,” *J. Teknol. Inf. dan Multimed.*, vol. 1, no. 1, pp. 8–13, May 2019.
- [5] J. Riady, H. N. Palit, and J. Andjarwirawan, “Aplikasi E-Learning Berbasis Progressive Web App Pada Apologetika Indonesia,” Surabaya, 2019.
- [6] A. Kurniawan, I. S. Areni, and A. Achmad, “Implementasi Progressive Web Application pada Sistem Monitoring Keluhan Sampah Kota Makassar,” *J. Penelit. Enjiniring, Fak. Tek. Univ. Hasanuddin*, vol. 21, no. 2, pp. 34–38, Nov. 2017, doi: 10.25042/jpe.112017.05.
- [7] Nurwanto, “Penerapan Progressive Web Application (PWA) pada E-Commerce,” *Techno.COM*, vol. 18, no. 3, pp. 227–235, Aug. 2019, doi: 10.33633/tc.v18i3.2400.
- [8] G. Matiini, R. Setiyadi, A. Setiawan, and M. Ramli, “Pengembangan Aplikasi Progressive Web Application (PWA) Untuk Pembelajaran dan Evaluasi Kelas English Grammar Online Course,” *J. Pendidik. Edutama*, vol. 8, no. 2, pp. 163–176, Jul. 2021, [Online]. Available: <http://ejurnal.ikipgribojonegoro.ac.id/index.php/JPE>
- [9] D. Haryanto and Z. R. S. Elsi, “Analisis Performance Progressive Web Apps Pada Aplikasi Shopee,” *J. Ilm. Inform. Glob.*, vol. 12, no. 2, pp. 106–111, Dec. 2021.
- [10] A. Mu’in, Amri, and Anwar, “Monitoring Sistem Keluhan Mahasiswa Menggunakan Progressive Web Application Pada Politeknik Negeri Lhokseumawe,” *J. Artif. Intell. Softw. Eng.*, vol. 1, no. 1, pp. 23–30, 2021, Accessed: Jun. 04, 2022. [Online]. Available: <http://e-jurnal.pnl.ac.id/JAISE/article/view/2218>
- [11] G. L. Dewi, S. Tjandra, and Ricardo, “Pemanfaatan Progressive Web Apps Pada Web Akuntansi,” *Teknika*, vol. 9, no. 1, pp. 38–47, Jul. 2020, doi: 10.34148/teknika.v9i1.252.
- [12] Web Dev, “PWA audits”, Accessed: Jun. 28, 2022. [Online]. Available: <https://web.dev/lighthouse-pwa/>
- [13] Dennis Sheppard, *Beginning Progressive Web App Development: Creating a Native App Experience on the Web*. Illinois, Amerika Serikat: Apress, 2017.
- [14] D. V Karpagam, P. R, L. R, and P. S, “Performance Enhancement Of Webpage Using Progressive Web App Features,” *Int. J. Innov. Res. Adv. Eng.*, vol. 4, no. 3, pp. 97–103, Mar. 2017, Accessed: Jun. 04, 2022. [Online]. Available: https://www.academia.edu/32134686/PERFORMANCE_ENHANCEMENT_OF_WEBPAGE_USING_PROGRESSIVE_WEB_APP_FEATURES
- [15] A. A. Kurniawan, “Analisis Performa Progressive Web Application (PWA) Pada Perangkat Mobile,” *J. Ilm. Inform. Komput.*, vol. 25, no. 1, pp. 18–31, Apr. 2020, doi: 10.35760/ik.2020.v25i1.2510.