Information System Journal (INFOS)
Vol. 7, No. 2, November 2024, pp. 104-117
E-ISSN : 2655-142X , P-ISSN : 2655-190X, DOI:https://doi.org/10.24076/infosjournal.2024v7i02.1986

ANALISIS DAN IMPLEMENTASI *PROGRESSIVE WEB APP* (PWA) SERTA FITUR NOTIFIKASI PADA SISTEM INFORMASI PENDAFTARAN *VOLUNTEER VOLHUB*

Hafid Hanifan [1)], **Ika Nur Fajri** [2)]
[1)2)] *Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Amikom Yogyakarta*

email: *hafidhaa@students.amikom.ac.id* [1)], *fajri@amikom.ac.id* [2)]

INFO ARTIKEL

Riwayat Artikel:

Diterima November, 2024
Revisi November, 2024
Terbit November, 2024

ABSTRAK

Perkembangan internet di Indonesia mendorong transformasi digital dalam berbagai aspek, termasuk kegiatan *volunteer*. Namun, tingkat partisipasi pemuda mengalami penurunan akibat akses dan mekanisme yang kurang memadai. *Website VolHub* dikembangkan untuk memfasilitasi pendaftaran *volunteer* secara efisien dan transparan, tetapi masih menghadapi tantangan dalam keandalan sistem. *Progressive Web App* (PWA) menjadi solusi untuk meningkatkan reliabilitas *VolHub* dengan fitur seperti akses *offline*, instalasi langsung dari *browser*, dan *push notifications*. Penelitian ini mengimplementasikan PWA pada *VolHub* menggunakan metode pengembangan berbasis *Agile*. Hasil menunjukkan peningkatan pengalaman pengguna, dengan akses lebih cepat dan stabil tanpa koneksi internet. Kesimpulannya, penerapan PWA dapat meningkatkan keandalan *website VolHub* serta mendorong partisipasi pemuda dalam kegiatan *volunteer*.

Kata Kunci :
Volunteer; *Progressive Web App*; *PWA*; *Service Worker*;

ABSTRACT

The rapid growth of the internet in Indonesia has driven digital transformation in various sectors, including volunteer activities. However, youth participation has declined due to inadequate access and mechanisms. VolHub is a website designed to facilitate volunteer registration efficiently and transparently but still faces reliability challenges. Progressive Web App (PWA) is a solution to enhance VolHub's reliability by enabling offline access, direct installation from browsers, and push notifications. This research implements PWA on VolHub using an Agile-based development approach. The results indicate improved user experience, with faster and more stable access even without an internet connection. In conclusion, PWA implementation enhances VolHub's reliability and encourages youth participation in volunteer activities.

***Keywords*:**
Volunteer; *Progressive Web App*; *PWA*; *Service Worker*;

Penulis Korespondensi:

Hafid Hanifan
Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Amikom Yogyakarta

Email:
hafidhaa@students.amikom.ac.id

1. PENDAHULUAN

Perkembangan teknologi dan internet di Indonesia semakin meningkat seiring dengan berjalannya waktu. Penelitian yang dilakukan oleh Asosiasi Penyelenggara Jasa Internet Indonesia (APJII) menunjukkan bahwa 92,21% masyarakat Indonesia menggunakan internet untuk mengakses informasi atau berita dengan 89,04% diantaranya menggunakan perangkat *mobile* [1]. Hal tersebut menunjukkan bahwa teknologi memiliki dampak yang sangat signifikan dalam berbagai aspek kehidupan, termasuk kegiatan sosial seperti *volunteer*. Teknologi secara tidak langsung memperluas cakupan dan efektivitas *volunteer* dengan mendukung proses pembentukan, pengorganisasian, serta pelaksanaan kegiatan [2]. Meskipun perkembangan teknologi sudah mempengaruhi kegiatan *volunteer*, namun tingkat partisipasi pemuda di Indonesia justru mengalami penurunan. Hal ini berdasarkan data dari Badan Perencanaan Pembangunan Nasional (Bappennas) yang

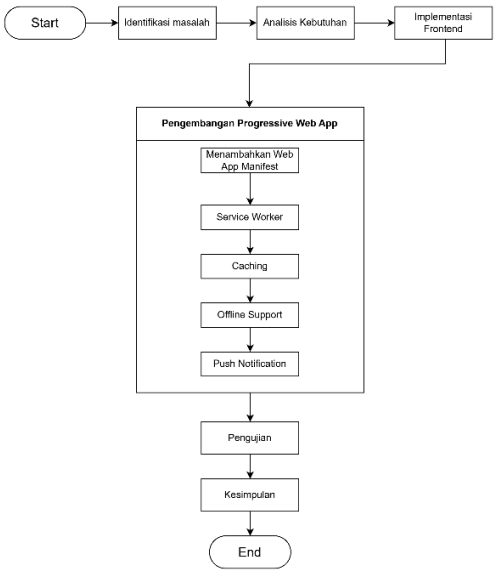
menyatakan tahun 2021 memiliki angka partisipasi terendah dengan jumlah 70,49% dibandingkan dengan tahun 2015 dan 2018 yang menyentuh angka 81,97% dan 81,36%. Penurunan angka tersebut salah satunya dikarenakan akses dan mekanisme yang belum memadahi [3]. Salah satu solusi untuk permasalahan tersebut yaitu hadirnya *website VolHub* yang bertujuan untuk memudahkan calon *volunteer* dalam melakukan pendaftaran secara efisien dan transparan. Namun keandalan (reliabilitas) pada *website VolHub* masih perlu ditingkatkan untuk memberikan pengalaman pengguna yang lebih baik.

Progressive Web App pertama kali diperkenalkan oleh insinyur *Google*, Alex Russell, pada Juni 2015 [4] [5] [6] [7]. *Progressive Web App* (PWA) merupakan jenis *website* yang memiliki pengalaman pengguna mirip dengan aplikasi *native* [8] [9] [10]. PWA memiliki beberapa kelebihan diantaranya dapat diakses meskipun tanpa koneksi internet, tidak perlu diinstal dari *App Store* atau *Playstore* serta memiliki fitur *push notifications* [11] [12] [13] [14]. Beberapa perusahaan besar seperti *Twitter*, *Pinterest*, dan *Forbes* telah berhasil mengimplementasikan PWA. *Twitter Lite* mengalami peningkatan keterlibatan pengguna serta pengurangan penggunaan data hingga 70% dibandingkan aplikasi *native*. PWA *Pinterest* meningkatkan keterlibatan pengguna sebesar 60% dan pendapatan iklan hingga 44%, sementara *Forbes* mencatat peningkatan sesi per-pengguna sebesar 43% serta peningkatan visibilitas iklan sebesar 20% [15]. Selain itu, penelitian yang dilakukan oleh Rifaldi Kusnawan menunjukkan bahwa implementasi *Progressive Web App* (PWA) dapat meningkatkan performa *website* secara signifikan. Peningkatan ini terjadi karena teknologi *caching* yang digunakan dalam *Progressive Web App* memungkinkan *website* untuk menyimpan data penting secara lokal, sehingga mengurangi waktu muat halaman dan ketergantungan pada koneksi internet. Dengan demikian, pengguna dapat mengakses halaman lebih cepat, bahkan dalam kondisi jaringan yang tidak stabil. Selain itu, fitur *caching* juga membantu mengoptimalkan efisiensi penggunaan *bandwidth*, sehingga mengurangi beban *server* dan meningkatkan pengalaman pengguna secara keseluruhan [7].

Pada *website VolHub* pengembangan *Progressive Web App* bertujuan untuk meningkatkan keandalan (reliabilitas) sistem dalam proses pendaftaran *volunteer*, khususnya dalam aspek performa. Salah satu keunggulan utama *Progressive Web App* yang diterapkan dalam penelitian ini adalah kemampuannya untuk menyediakan akses yang tetap optimal meskipun dalam kondisi jaringan yang tidak stabil atau bahkan tanpa koneksi internet. Dengan memanfaatkan fitur *service worker*, *website VolHub* dapat menyimpan *cache* halaman dan data penting, sehingga hal ini menjadi solusi atas kendala yang sering dihadapi calon *volunteer* yang memiliki keterbatasan akses internet. Selain itu, dengan adanya *Push Notification*, pengguna akan mendapatkan pemberitahuan langsung terkait program *volunteer* atau status pendaftaran, sehingga akan meningkatkan efektivitas komunikasi antara *platform* dan pengguna. Diharapkan dengan implementasi *Progressive Web App* pada *website VolHub* dapat meningkatkan performa pada *website*, dimana hal tersebut dapat meningkatkan keandalan (reliabilitas) dan memberikan pengalaman pengguna yang lebih baik.

2. METODOLOGI PENELITIAN

Tahapan pengembangan dan penelitian disajikan dalam bentuk alur sebagaimana yang dapat dilihat pada Gambar 1 berikut.



Gambar 1. Alur Penelitian.

Information System Journal (INFOS)
Vol. 7, No. 2, November 2024, pp. 104-117
E-ISSN : 2655-142X , P-ISSN : 2655-190X, DOI:https://doi.org/10.24076/infosjournal.2024v7i02.1986

2.1 Identifikasi Masalah

Identifikasi masalah berfokus pada keandalan PWA dan fitur notifikasi *VolHub*. Teknologi berperan dalam mempermudah pencarian serta pengelolaan program *volunteer*, namun akses dan mekanisme pendaftaran yang kurang memadai menurunkan partisipasi. Keandalan *website* perlu ditingkatkan untuk pengalaman pengguna yang lebih baik, serta diperlukan fitur pendukung seperti *push notifications*.

2.2 Analisis Kebutuhan

Analisis kebutuhan *VolHub* mencakup aspek fungsional, non-fungsional, dan teknis untuk memastikan sistem berjalan optimal. Kebutuhan fungsional meliputi desain responsif, dukungan PWA, akses *offline* melalui *caching Service Worker*, dan notifikasi *real-time* terkait status pendaftaran. Kebutuhan non-fungsional mencakup waktu muat kurang dari 3 detik, keamanan *HTTPS*, pembaruan *cache* otomatis, dan kompatibilitas dengan berbagai *browser*. Secara teknis, pengembangan PWA menggunakan *JavaScript* dengan *Service Worker* untuk *caching* dan *push notification* guna memberikan informasi terbaru kepada pengguna.

2.3 Implementasi *Frontend*

Bagian ini membahas proses penerapan antarmuka pengguna (UI) berdasarkan analisis kebutuhan yang telah dilakukan. Implementasi *frontend* mencakup pengembangan tampilan dan fungsionalitas menggunakan teknologi yang mendukung *Progressive Web App* (PWA).

2.4 Pengembangan *Progressive Web App*

Pengembangan PWA mencakup beberapa langkah utama. Pertama, pembuatan *Web App Manifest* dalam format *JSON* untuk mendefinisikan *metadata* aplikasi. Kedua, penambahan *Service Worker* sebagai komponen utama yang mendukung berbagai fitur PWA. Ketiga, pengaturan *caching* untuk menyimpan *file* statis secara lokal. Keempat, implementasi *offline capability* dengan *Service Worker* yang menangani permintaan melalui *event fetch*. Terakhir, penambahan *push notification* untuk mengirim pemberitahuan *real-time* ke pengguna.

2.5 Pengujian

Tahap penguijan dalam proses pengembangan PWA merupakan langkah penting untuk memastikan aplikasi berfungsi sesuai harapan. Evaluasi dilakukan dengan menggunakan metode pengujian berbasis *Lighthouse Audit*, yang berfokus pada aspek *Performance*. Pengujian dilakukan untuk mengukur waktu pemuatan halaman, responsivitas, serta efisiensi *caching* yang memungkinkan akses lebih cepat dan penggunaan *offline*. Pengujian ini dilakukan dengan menjalankan *Lighthouse* pada halaman-halaman utama, seperti *Home*, *Activity*, *Partner*, *Detail Partner*, *Profile*, *Login*, dan *Register* guna mendapatkan skor yang mencerminkan kecepatan dan keandalan sistem.

Hasil evaluasi ini kemudian dianalisis dengan membandingkan skor yang diperoleh sebelum dan sesudah optimalisasi PWA. Justifikasi dari hasil evaluasi dilakukan dengan mengamati perbedaan waktu muat, interaktivitas, serta penggunaan sumber daya. Jika skor *Performance* meningkat setelah penerapan PWA, maka dapat disimpulkan bahwa optimasi telah berhasil meningkatkan efisiensi dan keandalan *website*. Sebaliknya, jika tidak ada peningkatan yang signifikan, maka diperlukan perbaikan lebih lanjut, seperti optimasi gambar, pengurangan *render-blocking resources*, atau peningkatan strategi *caching*.

2.6 Kesimpulan

Tahap akhir ini merangkum seluruh hasil analisis dan evaluasi ke dalam temuan utama. Kesimpulan dibuat untuk menjawab permasalahan awal, serta mengevaluasi keefektifan PWA dalam konteks aplikasi yang dikembangkan.

3. HASIL DAN PEMBAHASAN

3.1. Implementasi *Frontend*

Pada bagian *frontend*, pengembangan dilakukan dengan menggunakan *framework Tailwind CSS* untuk memastikan tampilan yang responsif dan modern. Implementasi mencakup berbagai halaman yaitu halaman *Home*, *Activity*, *Partner*, *Detail Partner*, *Profile*, *Login* dan *Register*. Berikut merupakan rincian dari setiap implementasi yang dilakukan.

a. Implementasi Halaman *Home*

Gambar 2. Tampilan Halaman *Home*.

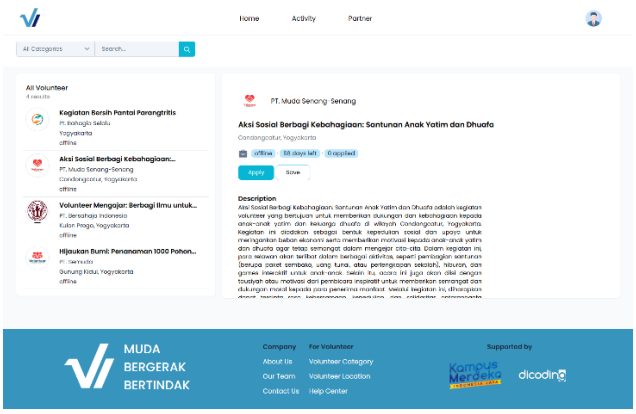
Bagian pertama dari halaman *home* menampilkan *hero section* yang berisikan gambar dan *tagline* utama dari *website VolHub* yaitu *"Make Meaningful Connections Through Volunteering"*, diikuti dengan deskripsi singkat mengenai *platform* dan tombol *Get Started* yang mendorong pengguna mengeksplorasi program *volunteer* yang tersedia.

Selanjutnya, terdapat *section* yang menjelaskan kemudahan registrasi pada platform *VolHub*. *Section* ini disusun dengan desain minimalis yang mencakup tiga keunggulan utama, yaitu *Effortless*, *Simple*, dan *Impactful*, masing-masing dilengkapi dengan ikon dan deskripsi singkat untuk memperjelas manfaat yang diperoleh pengguna dalam menggunakan *platform VolHub*.

Di bagian berikutnya, halaman *Home* menampilkan daftar peluang *volunteer* dalam bentuk kartu informasi yang memuat judul program, organisasi penyelenggara, deskripsi singkat, lokasi kegiatan, jumlah pendaftar, jangka waktu pendaftaran, dan sistem kegiatan.

Bagian *footer* halaman menampilkan identitas *VolHub* dengan tagline "Muda, Bergerak, Bertindak", diikuti dengan tautan navigasi ke berbagai halaman penting. Selain itu, terdapat dukungan dari program Kampus Merdeka dan Dicoding, yang ditampilkan dalam bentuk logo pada bagian bawah halaman.

b. Implementasi Halaman *Activity*

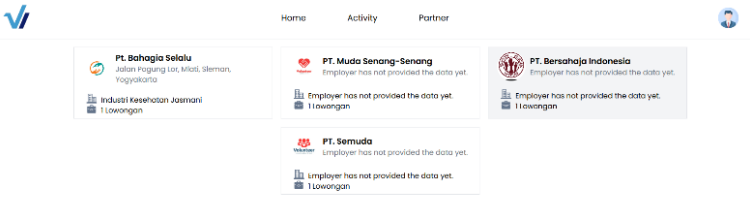


Gambar 3. Tampilan Halaman *Activity*.

Information System Journal (INFOS)
Vol. 7, No. 2, November 2024, pp. 104-117
E-ISSN : 2655-142X , P-ISSN : 2655-190X, DOI:https://doi.org/10.24076/infosjournal.2024v7i02.1986

Berdasarkan Gambar 3, halaman *activity* digunakan untuk menampilkan berbagai lowongan kegiatan *volunteer* yang ada. Halaman ini dibagi menjadi dua bagian. Pada bagian kiri, digunakan untuk menampilkan informasi singkat dari kegiatan. Sedangkan pada bagian kanan, digunakan untuk menampilkan detail informasi seperti PT penyelenggara, nama kegiatan, lokasi, dll.

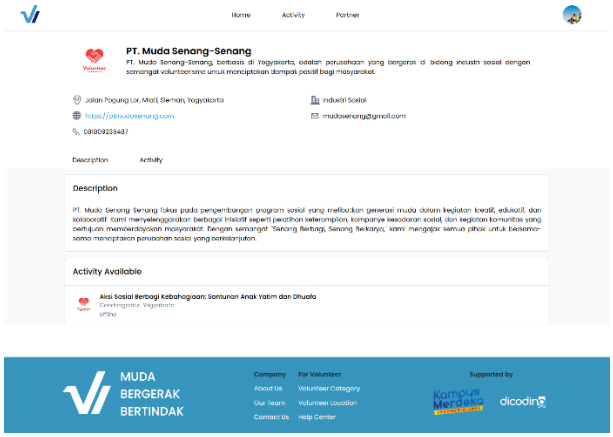
c. Implementasi Halaman *Partner*



Gambar 4. Tampilan Halaman *Partner*.

Halaman *partner* digunakan untuk menampilkan *employer* yang menyediakan berbagai macam kegiatan *volunteer*. Halaman ini terdiri dari beberapa *card* yang menampilkan informasi singkat dari *employer* seperti logo, nama *employer*, bidang industri, serta jumlah lowongan yang ada.

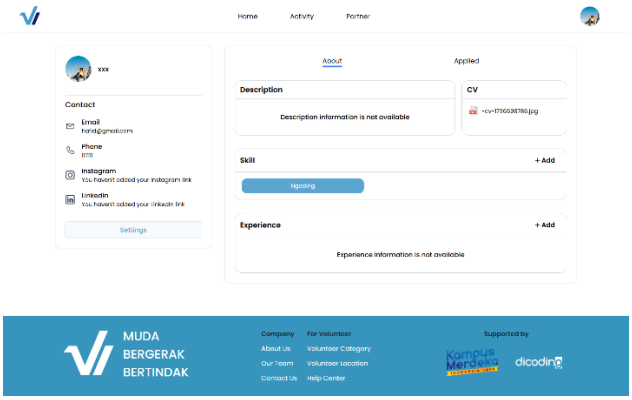
d. Implementasi Halaman Detail *Partner*



Gambar 5. Tampilan Halaman Detail *Partner*.

Halaman detail *partner* digunakan untuk menampilkan informasi detail informasi mengenai *employer*. Halaman ini menampilkan data seperti nama *employer*, bio, alamat, industri, dll.

e. Implementasi Halaman *Profile*



Gambar 6. Tampilan Halaman *Profile*.

Halaman ini digunakan untuk menampilkan informasi mengenai pengguna yang sedang *login*. Halaman ini dibagi menjadi dua bagian, yaitu untuk menampilkan kontak dan profil pengguna. Selain itu, halaman ini juga digunakan untuk melihat status pendaftaran yang telah dilakukan pengguna.

3.2. Pengembangan *Progressive Web App* (PWA)

Proses pengembangan *Progressive Web App* pada website VolHub memiliki beberapa tahapan untuk mencapai tujuan website yang dapat diandalkan. Dibawah ini merupakan rincian dari setiap proses yang ada.

a. Menambahkan *Web App Manifest*

Web App Manifest merupakan *file JSON* yang mendefinisikan tentang aplikasi PWA. *File* ini memungkinkan aplikasi *web* memiliki pengalaman seperti aplikasi *native* dengan mendukung pemasangan ke layar utama (*home screen*), *splash screen*, ikon aplikasi dan tampilan yang serupa dengan aplikasi *native*. Berikut merupakan *Manifest file* dari *website VolHub*.

```
{
    "name": "VolHub",
    "short_name": "VolHub",
    "start_url": "/",
    "display": "standalone",
    "background_color": "#ffffff",
    "theme_color": "#4a90e2",
    "icons": [
        {
            "src": "/images/icons/volhub-72x72.png",
            "sizes": "72x72",
            "type": "image/png"
        },
        {
            "src": "/images/icons/volhub-96x96.png",
            "sizes": "96x96",
            "type": "image/png"
        },
        {
            "src": "/images/icons/volhub-192x192.png",
            "sizes": "192x192",
            "type": "image/png"
        }
    ]
}
```

Gambar 7. *Manifest file*.

Properti *name* digunakan untuk menentukan nama lengkap aplikasi yang akan ditampilkan pada *launcher* atau daftar aplikasi, sedangkan *short_name* digunakan ketika ruang tampilan terbatas. Properti *start_url* berfungsi untuk menentukan halaman awal ketika website diakses, yaitu halaman utama. Selanjutnya, *background_color* dan *theme_color* digunakan untuk memberikan warna pada latar belakang dan UI tertentu seperti status bar pada perangkat Android.

Pada bagian *icons*, berisikan daftar ikon yang digunakan dalam berbagai ukuran layar dan resolusi perangkat. Setiap ikon memiliki properti *src* yang merujuk pada lokasi dari ikon yang digunakan, sizes yang menentukan dimensi ikon dalam piksel, serta *type* yang menunjukkan format dari ikon yang digunakan.

b. Service Worker

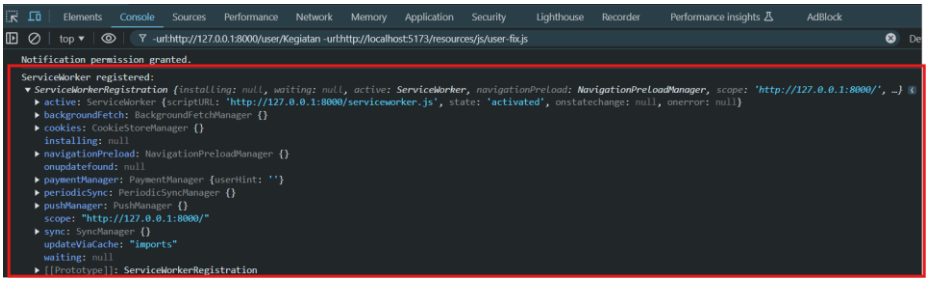
Service Worker berperan penting dalam proses pengembangan *Progressive Web App* yang digunakan untuk menghubugkan antara *client-side* dan *server-side*. Setiap permintaan akan melalui *Service Worker* sebelum diteruskan ke server. Begitu juga sebaliknya, *Service Worker* akan menyimpan data dari server ke dalam *cache*, sehingga website dapat diakses ketika tidak ada jaringan.

```
// Install Service Worker
if ("serviceWorker" in navigator) {
    window.addEventListener("load", () => {
        navigator.serviceWorker
            .register("/serviceworker.js")
            .then((registration) => {
                console.log("ServiceWorker registered: ", registration);
            })
            .catch((registrationError) => {
                console.log(
                    "ServiceWorker registration failed: ",
                    registrationError
                );
            });
    });
}
```

Gambar 8. *Service Worker*.

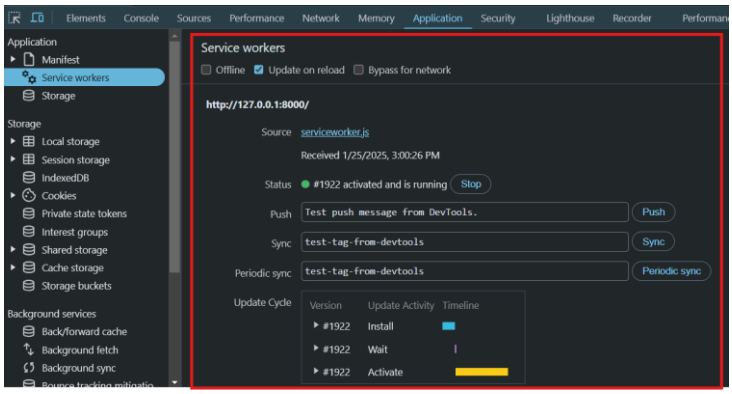
Ketika halaman website dimuat, *(window.addEventListener("load", ...)),* kode akan memerika dukungan terhadap *Service Worker*. Jika *browser* mendukung, maka *navigator.serviceWorker.register("/serviceworker.js")* akan dijalankan untuk mendaftarkan *file serviceworker.js* sebagai *Service Worker* pada *website*.

Information System Journal (INFOS)
Vol. 7, No. 2, November 2024, pp. 104-117
E-ISSN : 2655-142X , P-ISSN : 2655-190X, DOI:https://doi.org/10.24076/infosjournal.2024v7i02.1986



Gambar 9. Registrasi *Service Worker*.

Jika *Service Worker* berhasil di-*install*, maka pada *DevTools* pada *tab console* akan menunjukkan keterangan bahwa *Service Worker* berhasil di-*install*. Kemudian pada *tab Application* akan menunjukkan bahwa *Service Worker* akan menunjukkan status aktif dan menggunakan *source serviceworker.js* seperti pada Gambar 10.



Gambar 10. Tampilan *Service Worker* pada *DevTools*.

c. Caching

Proses selanjutnya setelah *Service Worker* berhasil diinstalasi pada *browser* adalah proses *Caching*. *Website VolHub* memanfaatkan *Cache Storage* untuk penyimpanan data sementara untuk nantinya ditampilkan pada sisi pengguna. Hal ini bertujuan untuk mengurangi latensi antara pengguna dan *server*. Rincinan proses pengembangan *caching* pada *website VolHub* akan dijabarkan pada penjelasan dibawah ini.

```
const CACHE_NAME = "volhub-cache-v1";
const urlsToCache = [
    "/",
    "/build/assets/app-BuWhUq6L.css",
    "/build/assets/app-C-NKBMM9.js",
    "/build/assets/user-fix-BhwnjqK6.js",
    "/build/assets/mitra-fix-CFwqNUXL.js",
];
```

Gambar 11. Implementasi *Cache*.

Pengembangan *cache* dimulai dengan memberi nama pada *cache* dengan cara membuat *variabel CACHE_NAME* dengan *name volhub-cache-v1*. Selain itu, *urlsToCache* berisi daftar *file* yang akan disimpan pada *cache*, termasuk halaman utama serta beberapa file *CSS* dan *Javascript* yang dihasilkan oleh *TailwindCSS* ketika proses *build*.

```
self.addEventListener("install", (event) => {
    event.waitUntil(
        caches.open(CACHE_NAME).then((cache) => {
            return cache.addAll(urlsToCache);
        })
    );
});
```

Gambar 12. *Install Cache*.

Event listener "*install*" akan dipicu ketika *Service Worker* pertama kali diinstalasi pada *browser*. Dalam *event* ini, metode *caches.open(CACHE_NAME)* digunakan untuk membuat atau membuka *cache* yang telah ditentukan, kemudian *cache.addAll(urlsToCache)* akan menyimpan *file-file* yang terdapat pada daftar *urlsToCaches* agar dapat digunakan ketika *offline*.

```
self.addEventListener("fetch", (event) => {
    event.respondWith(
        caches.match(event.request).then((response) => {
            return response || fetch(event.request);
        })
    );
});
```

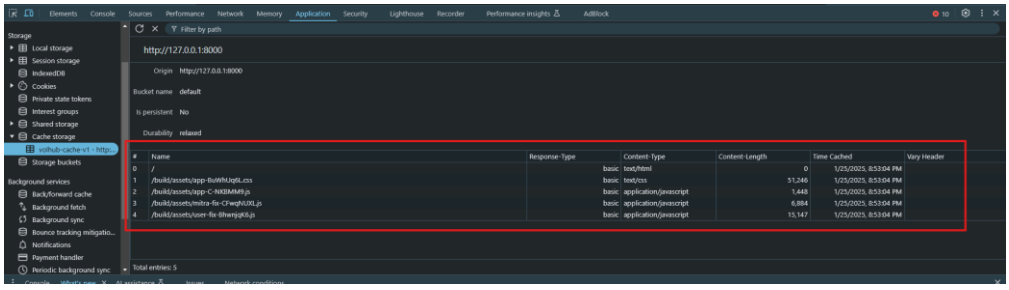
Gambar 13. Permintaan *Fetch*.

Kemudian *event listener* "*fetch*" digunakan untuk menangani permintaan jaringan. Ketika terdapat permintaan pada suatu sumber daya, *Service Worker* akan mencoba memeriksa *cache* terlebih dahulu. Jika sumber daya yang diminta terdapat pada *cache*, maka akan langsung diteruskan atau ditampilkan pada sisi pengguna. Namun, jika tidak ditemukan, maka permintaan akan diteruskan ke *server* melaui *(fetch(event.request))*. Hal ini bertujuan untuk menyediakan *file* yang akan ditampilkan pada sisi pengguna ketika *website* diakses tanpa menggunakan jaringan.

```
self.addEventListener("activate", (event) => {
    event.waitUntil(
        caches.keys().then((cacheNames) =>
            Promise.all(
                cacheNames.map((cache) => {
                    if (cache !== CACHE_NAME) {
                        return caches.delete(cache);
                    }
                })
            )
        )
    );
});
```

Gambar 14. Aktivasi *Cache*.

Terakhir, *event listener* "*activate*" digunakan untuk membersihkan *cache* lama yang sudah tidak diperlukan. Saat *Service Worker* diperbarui ke versi yang berbeda, semua *cache* yang tidak sesuai dengan *CACHE_NAME* akan dihapus melalui *caches.delete(cache).* Proses terakhir ini memastikan bahwa *website* menggunakan *cache* terbaru saja, sehingga meminimalisir *error* pada *website*.



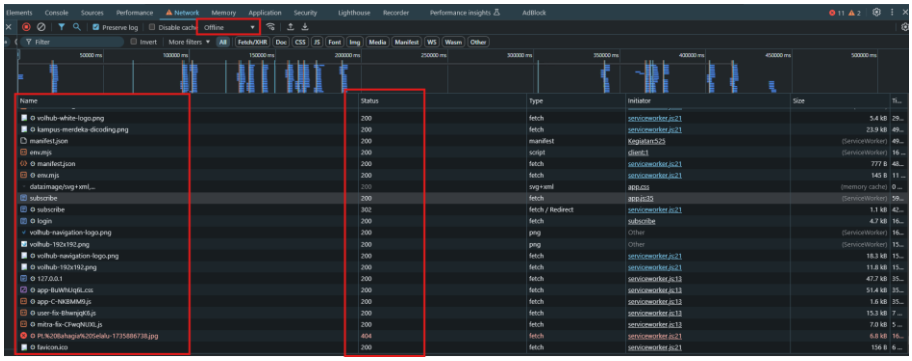
Gambar 15. Bukti *Cache* pada *DevTools*.

Berdasarkan hasil *Cache Storage* pada Gambar 15., yang ditampilkan pada *tab Application* pada *DevTools*, dapat disimpulkan bahwa *Service Worker* telah berhasil menyimpan beberapa aset penting dari *website VolHub* ke dalam *cache* bernama "*volhub-cache-v1*". Beberapa *file* yang berhasil di-*cache* meliputi halaman utama (*"/"*), *file CSS* utama (*app-BuWhUq6L.css*), serta beberapa *file JavaScript* seperti *app-C-NKBM9.js*, *mitra-fix-CFwqNUXL.js*, dan *user-fix-BhwnjqK6.js*.

d. Offline Support

Setelah proses implementasi *Service Worker* dan *Caching*, *website* harus dipastikan dapat diakses dalam keadaan *offline* atau tanpa jaringan. *DevTools* menjadi alat yang digunakan untuk mengecek status permintaan ketika *website* diakses dalam keadaan *offline*.

Information System Journal (INFOS)
Vol. 7, No. 2, November 2024, pp. 104-117
E-ISSN : 2655-142X , P-ISSN : 2655-190X, DOI:https://doi.org/10.24076/infosjournal.2024v7i02.1986



Gambar 16. Bukti *Offline Support*.

Berdasarkan Gambar 16., *DevTools* dari bawaan *browser* dapat digunakan sebagai alat pengecekan bahwa *website* tersedia dalam kondisi *offline* atau dapat diakses tanpa adanya jaringan. Memilih *mode Offline* pada *tab Network* bertujuan untuk mensimulasikan kondisi tanpa koneksi internet. Setelah *website* dimuat ulang, *resource* yang diperlukan (seperti gambar, *file*, *Javascript, CSS*, dan lainnya) akan memiliki status 200 OK. Hal ini membuktikan bahwa *website VolHub* tersedia dan dapat diakses tanpa jaringan.

e. Push Notifications

Selain memungkinkan *website* dapat diakses secara *offline*, *Progressive Web App* juga mendukung notifikasi melalui *Push Notification*, yang sangat bemanfaat untuk meningkatkan interaksi dengan pengguna secara *real-time*. Salah satu cara untuk mengimplementasikan fitur ini adalah dengan memanfaatkan *library Minishlink\WebPush\WebPush, Minishlink\WebPush\Subscription*, serta menggunakan *Vapid (Voluntary Application Server Identification)* sebagai metode autentikasi dalam pengiriman notifikasi.

```
const csrfToken = document.querySelector('meta[name="csrf-token"]').content;
if ("Notification" in window && "serviceWorker" in navigator) {
    Notification.requestPermission().then((permission) => {
        if (permission === "granted") {
            console.log("Notification permission granted.");
            // Daftarkan service worker jika belum
            navigator.serviceWorker.ready.then((registration) => {
                registration.pushManager
                    .subscribe({
                        userVisibleOnly: true,
                        applicationServerKey: urlBase64ToUint8Array(
                            "BNHNnasVegKlFNPH3PNz-4X731y1zwPrzZ0ZkmB7hjp4zHeNzMBACHsbvkI4ttu_g8LN-Eh3HtCZ8CSeQD2eWF8"
                        ), // Ganti dengan kunci publik VAPID Anda
                    })
                    .then((subscription) => {
                        // Kirim subscription ke server untuk disimpan
                        return fetch("/user/subscribe", {
                            method: "POST",
                            headers: {
                                "Content-Type": "application/json",
                                "X-CSRF-TOKEN": csrfToken,
                            },
                            body: JSON.stringify(subscription),
                        });
                    })
                    .then(() => {
                        console.log("Subscription saved.");
                    })
                    .catch((err) => {
                        console.error("Failed to subscribe:", err);
                    });
            });
        } else {
            console.log("Notification permission denied.");
        }
    });
}
```

Gambar 17. *Script Subscription Notification*.

Proses dimulai ketika pengguna memberikan izin supaya *website* dapat mengirimkan notifikasi. Jika pengguna memberikan izin, browser akan membuat *subscription* baru menggunakan *pushManager.subscribe(),* yang mencakup informasi seperti *endpoint, keys.p256dh*, dan *keys.auth*. Kemudian data akan dikirimkan ke *backend* melalui *endpoint API* menggunakan metode *HTTP POST* untuk disimpan ke *database*.

```php
// Kirim push notification
        $subscriptions = $pendaftar->user->pushSubscriptions ?? [];
        if ($subscriptions->isEmpty()) {
            return redirect()->back()->with('message', 'Tidak ada subscription untuk dikirim notifikasi.');
        }

        $webPush = new WebPush([
            'VAPID' => [
                'subject' => 'mailto:example@example.com',
                'publicKey' => config('webpush.vapid.public_key'),
                'privateKey' => config('webpush.vapid.private_key'),
            ],
        ]);

        foreach ($subscriptions as $subscription) {
            // Buat objek Subscription
            $webPushSubscription = Subscription::create([
                'endpoint' => $subscription->endpoint,
                'publicKey' => $subscription->p256dh,
                'authToken' => $subscription->auth,
            ]);

            $payload = json_encode([
                'title' => 'Shortlist Notification',
                'body' => "Hello {$pendaftar->user->nama_user}, you have been shortlisted!",
            ]);

            // Tambahkan notifikasi ke antrean
            $webPush->queueNotification($webPushSubscription, $payload);
        }

        // Kirim semua notifikasi sekaligus
        $reports = $webPush->flush();

        // Periksa laporan
        foreach ($reports as $report) {
            $endpoint = $report->getRequest()->getUri()->__toString();

            if ($report->isSuccess()) {
                logger("Notification sent successfully to {$endpoint}.");
            } else {
                logger("Notification failed to {$endpoint}: {$report->getReason()}");
            }
        }
```

Gambar 18. *Script* Pengiriman Notifikasi.

Setelah data *subscription* dimasukkan ke *database*, data akan diambil melalui *$pendaftar->user->pushSubscriptions*. Jika tidak ada *subscription* yang ditemukan, proses dihentikan dengan pemberitahuan ke pengguna. Selanjutnya, *library Minishlink\WebPush\WebPush* diinisialisasi dengan konfigurasi *VAPID* yang mencakup kunci publik, kunci privat, dan subjek pengirim. Untuk setiap *subscription*, dibuat objek *Subscription* menggunakan data seperti *endpoint, publicKey*, dan *authToken*. *Payload* notifikasi yang berisi judul dan pesan personalisasi kemudian disiapkan dan ditambahkan ke antrean pengiriman melalui metode *queueNotification()*. Setelah semua notifikasi dimasukkan ke antrean, metode *flush()* dipanggil untuk mengirim notifikasi ke semua endpoint secara bersamaan.

3.3. Pengujian

Pengujian dilakukan untuk menganalisis *Performance* dan *Progressive Web App* menggunakan *Lighthouse* yang dapat diakses pada *Chrome Devtools*. Dikarenakan hasil audit dari *Lighthouse* hanya mengaudit satu halaman per proses, maka skor akhir setiap kategori dihitung sebagai rata-rata dari semua halaman yang diuji. Setiap kategori memiliki bobot nilai yang akan menjadi tolak ukur kualitas halaman berdasarkan warna indikator yang ditampilkan oleh *Lighthouse*.

Tabel 1. Skala Nilai.

Skor	Status	Warna Status
0-49	Kurang Baik	Merah
50-89	Cukup	Jingga
90-100	Sangat Baik	Hijau

Terdapat satu kategori dan lima sub kategori yang dijadikan tolak ukur pengujian. Hal ini berdasarkan pada hasil yang diperolah dari audit *Lighthouse* pada browser. Berikut merupakan rincian dari kategori dan sub kategori yang dimaksud:

1. *Performance,* yaitu proses evaluasi kecepatan dan efisiensi *website* saat dimuat oleh pengguna. Kategori ini juga merupakan akumulasi dari keseluruhan nilai sub kategori yang ada.
2. *First Contentful Paint (FCP),* yaitu waktu yang dibutuhkan untuk menampilkan elemen pertama pada layar.
3. *Largest Contentful Paint (LCP),* yaitu waktu yang diperlukan untuk memuat elemen terbesar pada halaman *website*, seperti gambar atau teks utama.
4. *Total Blocking Time (TBT),* yaitu durasi dimana interaktivitas halaman tertunda akibat proses pemuatan skrip yang berat.
5. *Cumulative Layout Shift (CLS),* yaitu seberapa banyak tata letak halaman berubah selama pemuatan, dimana hal tersebut mengganggu pengalaman pengguna.
6. *Speed Index,* yaitu seberapa cepat konten terlihat secara visual selama proses pemuatan *website*.

Information System Journal (INFOS)
Vol. 7, No. 2, November 2024, pp. 104-117
E-ISSN : 2655-142X , P-ISSN : 2655-190X, DOI:https://doi.org/10.24076/infosjournal.2024v7i02.1986

Selain itu, terdapat 3 skenario utama dalam proses pengujian yang dilakukan untuk mengevaluasi kinerja sistem dalam berbagai kondisi, yaitu:

1. Pengujian tanpa *cache*. Pengujian ini dilakukan tanpa menggunakan *cache*, shingga setiap permintaan data akan dimuat langsung dari *server*. Skenario ini merepresentasikan pengalaman pengguna yang baru mengakses *website* untuk pertama kali.
2. Pengujian dengan *cache*. Pada skenario ini, *cache* digunakan untuk menyimpan elemen-elemen halaman. Sehingga memungkinkan *website* dapat diakses dalam waktu yang lebih singkat.
3. Pengujian *offline* dengan *cache*. Pengujian ini mensimulasikan kondisi ketika pengguna mengakses *website* tanpa menggunakan koneksi internet, tetapi dengan *cache* yang sudah tersimpan pada *storage* browser. Skenario ini bertujuan untuk mengukur efektivitas implementasi *Progressive Web App* (PWA) dalam meyediakan akses *offline*.

Berikut merupakan hasil dari pengujian yang telah dilakukan pada setiap halaman berdasarkan kategori, sub kategori dan skenario yang telah ditentukan.

Tabel 2. Hasil Pengujian Tanpa *Cache*.

	Performance					
	First Contentful Paint (s)	***Largest Contentful Paint (s)***	***Total Blocking Time (s)***	***Cumulative Layout Shift (s)***	***Speed Index (s)***	**Skor *Performance* (Akumulasi)**
Home	1.0	21.4	0	0	1.0	73
Activity	0.9	1	0	0	0.9	97
Partner	1.0	1.1	0	0	1.0	97
Detail Partner	0.8	0.8	0	0	0.8	99
Profile	1.0	1.0	0	0	1.0	97
Login	0.7	0.7	0	0	0.8	99
Register	0.8	0.8	0	0	0.9	99
Rata-rata	0.89	3.83	0	0	0.91	94.43

Tabel 3. Pengujian dengan *Cache*.

	Performance					
	First Contentful Paint (s)	***Largest Contentful Paint (s)***	***Total Blocking Time (s)***	***Cumulative Layout Shift (s)***	***Speed Index (s)***	**Skor *Performance* (Akumulasi)**
Home	0.1	0.4	0	0	0.5	100
Activity	0.6	0.7	0	0	0.6	100
Partner	0.1	0.1	0	0	0.1	100
Detail Partner	0.6	0.6	0	0	0.6	100
Profile	0.1	0.1	0	0	0.1	100
Login	0.1	0.1	0	0	0.3	100
Register	0.1	0.1	0	0	0.3	100
Rata-rata	0.24	2.1	0	0	0.36	100

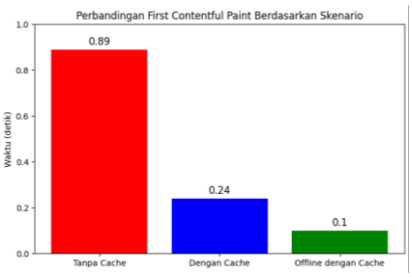
Tabel 4. Hasil Pengujian *Offline* Tanpa *Cache*.

	Performance					
	First Contentful Paint (s)	***Largest Contentful Paint (s)***	***Total Blocking Time (s)***	***Cumulative Layout Shift (s)***	***Speed Index (s)***	**Skor *Performance* (Akumulasi)**
Home	0.1	0.4	0	0	0.6	100
Activity	0.1	0.1	0	0	0.1	100
Partner	0.1	0.1	0	0	0.1	100
Detail Partner	0.1	0.1	0	0	0.1	100
Profile	0.1	0.1	0	0	0.1	100
Login	0.1	0.1	0	0	0.3	100
Register	0.1	0.1	0	0	0.3	100
Rata-rata	0.1	1.0	0	0	0.23	100

Tabel 2-4., merupakan hasil pengujian melalui tools *Lighthouse* yang terdapat pada browser *Chrome*. Pengujian ini dilakukan pada setiap halaman yang diuji, dengan hasil yang diperoleh dari masing-masing kategori dan subkategori dalam *Lighthouse*, khususnya pada kategori *Performance*. Masing-masing skor *Performance* memiliki skala 0-100. Berikut keterangan lebih detail mengenai hasil yang diperoleh setelah pengujian:

a. *First Contentful Paint (FCP)*

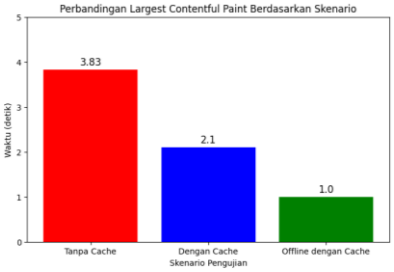
Berdasarkan Gambar 19, grafik ketika *website* diakses tanpa menggunakan *cache* menunjukkan waktu 0.89 detik. Hal tersebut dikarenakan *browser* harus mengambil semua aset dari server seperti *HTML, CSS, Javascript* dan gambar yang diperlukan. Pada skenario pengujian dengan menggunakan *cache*, grafik menunjukkan waktu 0.24 detik. Waktu tersingkat diperoleh ketika pengujian *offline* dengan menggunakan *cache* yang menunjukkan waktu 0.1 detik.



Gambar 19. Perbandingan FCP.

b. *Largest Contentful Paint (LCP)*

Berdasarkan Gambar 20., grafik menunjukkan bahwa skenario *offline* dengan *cache* menunjukkan waktu 1.0 detik. Hal tersebut membuktikan bahwa *Service Worker* berfungsi secara optimal dalam menyajikan elemen terbesar secara langsung dari *cache* tanpa perlu melakukan permintaan ke *server* dengan menggunakan jaringan.



Gambar 20. Perbandingan LCP.

Information System Journal (INFOS)
Vol. 7, No. 2, November 2024, pp. 104-117
E-ISSN : 2655-142X , P-ISSN : 2655-190X, DOI:https://doi.org/10.24076/infosjournal.2024v7i02.1986

c. *Total Blocking Time (TBT)*

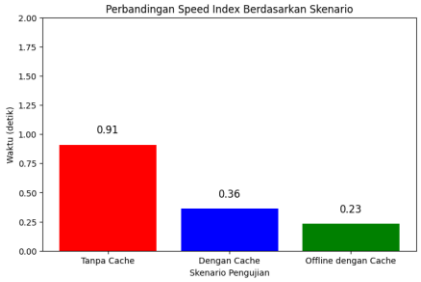
Berdasarkan pengujian yang telah dilakukan, nilai TBT menujukkan angka 0 dari setiap skenario pengujian yang dilakukan. Hal ini mengindikasikan bahwa selama proses pemutan halaman, tidak terdapat tugas berat (*long tasks*) yang menyebabkan pemblokiran interaksi pengguna dengan halaman yang sedang diakses.

d. *Cumulative Layout Shift (CLS)*

Pada pengujian yang dilakukan, semua nilai pengujian menunjukkan angka 0 dari setiap skenario pengujian yang telah dilakukan. Nilai 0 pada CLS menandakan bahwa tidak ada pergeseran tata letak yang mengganggu selama proses pemuatan halaman, yang berarti elemen-elemen pada halaman tetap stabil dan tidak berubah posisi secara tiba-tiba setelah mulai ditampilkan.

e. *Speed Index*

Berdasarkan Gambar 21., grafik menunjukkan nilai terendah terdapat pada skenario *offline* dengan *cache*. Hal ini menunjukkan bahwa *Service Worker* bekerja secara optimal dalam menyajikan aset yang telah tersimpan sebelumnya, sehingga halaman dapat ditampilkan dengan sangat cepat tanpa harus bergantung pada koneksi internet.



Gambar 21. Perbandingan *Speed Index*.

4. KESIMPULAN

Berdasarkan hasil pengujian, penerapan PWA pada *VolHub* terbukti meningkatkan performa dan pengalaman pengguna secara signifikan. Dalam skenario tanpa *cache*, nilai rata-rata *First Contentful Paint* (FCP) sebesar 0.89 detik, *Largest Contentful Paint* (LCP) 3.83 detik, *Speed Index* 0.91 detik, dan skor *Performance* 94.43. Setelah optimasi dengan *cache*, terjadi peningkatan performa dengan FCP menurun menjadi 0.24 detik, LCP 2.1 detik, *Speed Index* 0.36 detik, dan skor *Performance* meningkat menjadi 100. Pada skenario *offline* dengan *cache*, performa semakin optimal dengan FCP dan LCP hanya 0.1 detik serta skor *Performance* tetap 100, menunjukkan bahwa *VolHub* tetap dapat diakses dengan lancar tanpa koneksi internet.

Hasil evaluasi ini menunjukkan bahwa implementasi PWA berhasil meningkatkan kecepatan pemuatan halaman, mengoptimalkan *caching*, serta memastikan keandalan akses bahkan dalam kondisi *offline*. Selain itu, fitur notifikasi yang diterapkan memungkinkan pengguna untuk menerima informasi terbaru terkait program volunteer secara *real-time*, sehingga meningkatkan keterlibatan mereka dalam proses pendaftaran. Dengan demikian, penerapan PWA dan fitur notifikasi pada VolHub telah berhasil menjawab permasalahan aksesibilitas dan efisiensi sistem, serta dapat menjadi referensi bagi pengembang lain dalam mengadopsi teknologi serupa.

DAFTAR PUSTAKA

[1] G. P. Riyanto and Y. Pratomo, "APJII: Pengguna Internet di Indonesia Tembus 210 Juta pada 2022," Kompas Tekno, 10 Juni 2022. [Online]. Tersedia: https://tekno.kompas.com/read/2022/06/10/19350007/pengguna-internet-di-indonesia-tembus-210-juta-pada-2022. [Diakses: 02-Nov-2024]

[2] F. Demiroz and E. Akbas, "The Impact of Social Media on Disaster Volunteerism: Evidence from Hurricane Harvey," *J. Homel. Secur. Emerg. Manag.*, vol. 19, no. 2, pp. 205–243, 2022, doi: 10.1515/jhsem-2020-0077.

[3] N. Rosa, "Bappenas: Angka Partisipasi Pemuda Menurun dalam Pembangunan Indonesia Emas 2045," 2023. [Online]. Tersedia: https://www.detik.com/edu/edutainment/d-6704311/bappenas-angka-partisipasi-pemuda-menurun-dalam-pembangunan-indonesia-emas-2045. [Diakses: 02-Nov-2024].

[4] A. A. Kurniawan, "Analisis Performa Progressive Web Application (Pwa) Pada Perangkat Mobile," *J. Ilm. Inform. Komput.*, vol. 25, no. 1, pp. 18–31, 2020, doi: 10.35760/ik.2020.v25i1.2510.

[5] R. V. Rochim, A. Rahmatulloh, R. R. El-Akbar, and R. Rizal, "Performance Comparison of Response Time Native, Mobile and Progressive Web Application Technology," *Innov. Res. Informatics*, vol. 5, no. 1, pp. 36–43, 2023, doi: 10.37058/innovatics.v5i1.7045.
[6] C. Avinash Devarapalli, "Progressive Web App (PWA): Optimal Strategies & Challenges," *Int. J. Res. Eng. Sci. ISSN*, vol. 12, no. 3, pp. 174–180, 2024, [Online]. Available: www.ijres.org
[7] R. Kusnawan, "Implementasi Progressive Web App pada Aplikasi Manajemen Data Dropship," 2021.
[8] A. Amrullah, Y. Salim, and A. Rachman Manga, "Implementasi Progressive Web App Terhadap Aplikasi E-Commerce Sebagai Solusi Untuk Meningkatkan Kinerja Aplikasi Berbasis Web Informasi Artikel Abstrak," *Bul. Sist. Inf. dan Teknol. Islam*, vol. 2, no. 3, pp. 213–221, 2021.
[9] A. Prayitno, E. Hariyanto, and Suheri, "Perancangan Aplikasi Pengelolaan Keuangan Menggunakan Metode Progressive Web Apps (Studi Kasus : SDIT Zahra Asy Syifa Patumbak Deli Serdang)," *Bull. Inf. Technol.*, vol. 4, no. 1, pp. 9–14, 2023, doi: 10.47065/bit.v4i1.452.
[10] D. Nugraha, F. Anjara, and S. Faizah, "Comparison of Web Based and PWA in Online Learning," *Proc. 5th FIRST T1 T2 2021 Int. Conf. (FIRST-T1-T2 2021)*, vol. 9, pp. 201–205, 2022, doi: 10.2991/ahe.k.220205.035.
[11] S. Jodi and R. Amin, "Implementation of progressive web apps-based click profile on social media," *J. Ilmu Pengetah. dan Teknol. Komput.*, vol. 7, no. 1, pp. 23–28, 2021, [Online]. Available: https://doi.org/10.33480/jitk.v7i1.2354
[12] 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, p. 163, 2021, doi: 10.30734/jpe.v8i2.984.
[13] M. Bennervall, "A Comparison Of Progressive Web Apps And," 2024.
[14] O. Adetunji, C. Ajaegbu, N. Otuneme, and O. J. Omotosho, "Dawning of Progressive Web Applications (PWA): Edging Out the Pitfalls of Traditional Mobile Development," *Technol. Sci. Am. Sci. Res. J. Eng.*, vol. 68, no. 1, pp. 85–99, 2020, [Online]. Available: http://asrjetsjournal.org/
[15] C. Technology and F. Olowoniyi, "Design and implementation of a PWA for ordering taxi," no. May, 2023.