JIKO (JURNAL INFORMATIKA DAN KOMPUTER)
September 2024, Volume: **8**, No. **2** | Pages 404–418
doi: 10.26798/jiko.v8i2.1319
e-ISSN : 2477-3964 – p-ISSN : 2477-4413

ARTICLE

Pengelolaan Cache pada Aplikasi Pencatatan Penjualan Menggunakan Fuzzy Page Replacement Algorithm

Cache Management in a Sales Recording Application Using Fuzzy Page Replacement Algorithm

Bregsi Atingsari Julastri, Anggraini Puspita Sari,* dan Made Hanindia Prami Swari

Informatika, Fakultas Ilmu Komputer, Universitas Pembangunan Nasional "Veteran" Jawa Timur, Surabaya, Indonesia
*Penulis Korespondensi: anggraini.puspita.if@upnjatim.ac.id

(Disubmit 24-05-16; Diterima 24-06-03; Dipublikasikan online pada 24-09-05)

Abstrak

Tantangan yang dihadapi dalam pengembangan suatu website semakin beragam, khususnya untuk menyajikan sebuah website yang responsif dan sesuai dengan preferensi pengguna. Progressive Web App (PWA) merupakan teknologi web canggih yang menggabungkan keunggulan dari aplikasi mobile dengan kemudahan akses dari browser. Dengan fitur-fitur seperti offline access dan responsivitas pada berbagai perangkat melalui pemanfaatan web cache, PWA dapat memberikan pengalaman pengguna yang lebih baik dan interaktif. Penelitian ini bertujuan untuk mengembangkan aplikasi pencatatan penjualan berbasis PWA yang dioptimalkan menggunakan algoritma dengan pendekatan fuzzy, yaitu Fuzzy Page Replacement Algorithm (FPRA) dalam pengelolaan cache-nya. FPRA diterapkan untuk meningkatkan efisiensi cache melalui klasterisasi halaman berdasarkan tingkat kebaruan, frekuensi akses, dan tingkat referensi halaman menggunakan algoritma Fuzzy C-Means. Penelitian dilakukan dengan studi kasus pada aplikasi pencatatan penjualan, yang menggunakan metode pengembangan aplikasi Rapid Application Development (RAD). Hasil penelitian menunjukkan bahwa FPRA berhasil menurunkan waktu muat halaman secara signifikan hingga mencapai 21% lebih cepat yang diujikan dalam tujuh skenario, serta dapat meningkatkan cache hit ratio hingga mencapai 58% dibandingkan dengan algoritma FIFO dan LFU. Selain itu, penerapan FPRA pada service worker juga meningkatkan rata-rata skor performa aplikasi sebesar 15 poin, dari 56 menjadi 71, yang dievaluasi menggunakan alat pengujian Lighthouse. Dengan demikian, penerapan FPRA dalam pengelolaan cache terbukti dapat meningkatkan kinerja aplikasi.

Kata kunci: Progressive Web App; Manajemen Cache; Fuzzy Page Replacement Algorithm; Aplikasi Pencatatan Penjualan

Abstract

Challenges faced in website development are increasingly diverse, particularly in presenting a responsive website that aligns with user preferences. Progressive Web App (PWA) is an advanced web technology that combines the advantages of mobile applications with the accessibility of browsers. With features such as offline access and responsiveness across various devices through the utilization of web cache, PWAs can provide a better and more interactive user experience. This research aims to develop a PWA-based sales recording application optimized using a fuzzy approach algorithm, specifically the Fuzzy Page Replacement Algorithm (FPRA) for cache management. FPRA is applied to enhance cache efficiency by clustering pages based on recency, access frequency, and page reference levels using the Fuzzy C-Means algorithm. The research was conducted as a case study on a sales recording application, utilizing the Rapid Application Development (RAD) method. The results show that FPRA successfully reduced page load time significantly, achieving up to 21% faster loading in seven tested scenarios, and increased the cache hit ratio by up to 58% compared to FIFO and LFU algorithms. Additionally, implementing FPRA in the service worker also improved the average application performance score

How to Cite: B. A. Julastri *et al.*, "Pengelolaan Cache pada Aplikasi Pencatatan Penjualan Menggunakan Fuzzy Page Replacement Algorithm", *JIKO (JURNAL INFORMATIKA DAN KOMPUTER)*, Volume: **8**, No.**2**, Pages 404–418, September 2024, doi: 10.26798/jiko.v8i2.1319.

by 15 points, from 56 to 71, evaluated using the Lighthouse testing tool. Therefore, the application of FPRA in cache management has been proven to enhance application performance.

KeyWords: Progressive Web App; Cache Management; Fuzzy Page Replacement Algorithm; Sales Recording Application

1. Pendahuluan

Dalam sebuah bisnis penjualan, efisiensi proses pencatatan menjadi poin penting untuk diperhatikan. Digitalisasi pencatatan penjualan melalui sebuah sistem telah menjadi solusi yang umumnya digunakan untuk meningkatkan efisiensi pencatatan. Digitalisasi ini dapat menghemat waktu dan mengurangi kesalahan perhitungan saat perekapan hasil penjualan, memudahkan pengelolaan penjualan karena datanya dapat diakses dari mana saja selama terhubung ke sistem [1]. Namun, banyak tantangan yang dihadapi demi mengembangkan sistem yang efektif, seperti tantangan untuk memastikan teknologi yang digunakan dapat mengakomodasi kebutuhan pengguna, memberikan responsivitas yang baik, dan tetap berfungsi optimal meskipun dengan keterbatasan akses internet.

Salah satu metode baru dalam pengembangan sistem berbasis web adalah menerapkan paradigma aplikasi web progresif (*Progressive Web App*) atau biasa disingkat PWA. PWA merupakan pendekatan baru dalam pengembangan aplikasi *mobile* yang diperkenalkan oleh Google pada tahun 2015 dengan menggunakan teknologi web modern, seperti *web manifest* dan *service worker* [2]. Dengan adanya *service worker*, PWA telah membuktikan keandalannya dalam meningkatkan responsivitas sistem dengan mempercepat waktu muat halaman hingga 2 kali lebih cepat [3] dan memungkinkan sebuah website dapat diakses secara *offline* (tanpa terhubung ke internet) dengan memanfaatkan *cache* [4].

Manajemen *cache* memegang peranan penting dalam menentukan kecepatan dan efisiensi PWA. Dengan menyimpan data secara lokal pada perangkat pengguna, manajemen *cache* dapat mengurangi latensi akses dan meningkatkan responsivitas aplikasi. Namun, pengelolaan *cache* memerlukan strategi yang tepat, terutama dalam hal penggantian halaman (*page replacement*) untuk menentukan objek yang harus dihapus ketika *cache* penuh [5]. Salah satu metode menarik yang bisa diterapkan dalam pengelolaan *cache* ini adalah menggunakan algoritma penggantian halaman dengan pendekatan *fuzzy*, bernama *Fuzzy Page Replacement Algorithm* (FPRA).

FPRA merupakan algoritma pergantian halaman yang diusulkan oleh Davood Akbari Bengar, dkk. pada tahun 2020 dengan mempertimbangkan tingkat kepentingan data suatu halaman di dalam *cache* berdasarkan parameter tingkat kebaruan, frekuensi, dan tingkat referensi setiap halaman. FPRA mengadopsi algoritma klasterisasi, *Fuzzy C-Means* (FCM), untuk mengelompokkan halaman web. Algoritma FCM membuat halaman yang lebih mirip satu sama lain berada dalam satu klaster yang sama. Penerapan logika *fuzzy* membuat setiap klaster bisa menjadi prioritas, bisa juga menjadi bukan prioritas karena nilai keanggotaannya di antara 0 dan 1 [6]. Jumlah klaster adalah sepuluh persen dari jumlah maksimum *total frame cache*. Klaster yang memiliki anggota halaman terbaru, lebih sering diakses, dan memiliki tingkat referensi yang lebih kecil akan memiliki prioritas yang lebih tinggi untuk tetap disimpan dalam *cache*. Dari hasil penelitiannya, FPRA dapat meningkatkan *cache hit ratio* yang membuat *cache* dapat bekerja lebih efisien dibandingkan dengan enam algoritma terdahulu, seperti WRP, DWRP, MFU, LFU, LRU, dan FIFO [7].

Berdasarkan permasalahan dan hasil penelitian mengenai kinerja FPRA dalam pengelolaan *cache*, penelitian ini bertujuan untuk mengembangkan sebuah aplikasi pencatatan penjualan berbasis PWA serta mengoptimalkan kinerjanya melalui penerapan manajemen *cache* dengan algoritma penggantian halaman FPRA. Dengan memperhatikan karakteristik PWA dan kebutuhan bisnis dalam pencatatan penjualan, penelitian ini diharapkan dapat meningkatkan efisiensi dan responsivitas aplikasi.

2. Metode

2.1 Pengumpulan Data

Penelitian ini menggunakan studi kasus sebuah usaha ayam geprek yang belum memiliki sistem pencatatan penjualan secara digital. Oleh karena itu, perlu dilakukan pengumpulan data agar aplikasi yang dirancang dan dikembangkan bisa memenuhi kebutuhan pengguna akhir. Metode pengumpulan data yang

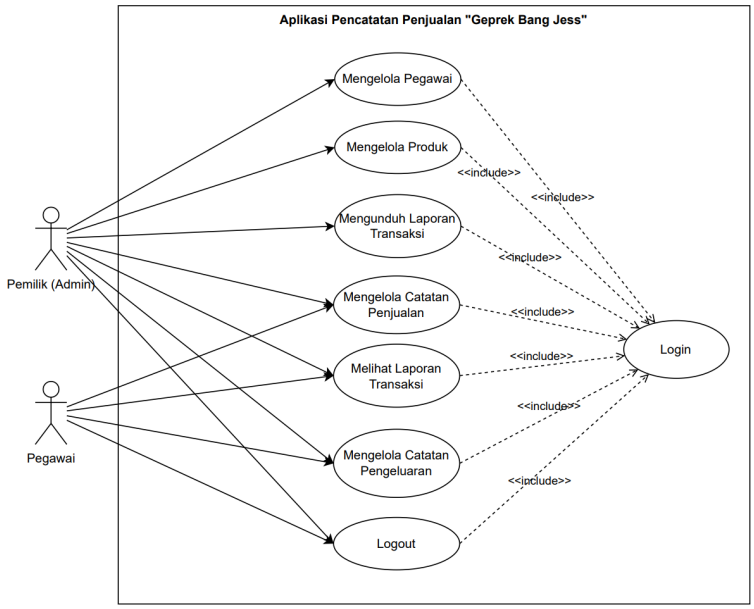
digunakan meliputi observasi, wawancara, dan studi literatur. Observasi dilakukan dengan mengamati langsung proses pencatatan penjualan di lokasi usaha tersebut guna memahami permasalahan yang ada. Wawancara semi terstruktur dengan pemilik usaha dengan tujuan untuk mengetahui kebutuhan aplikasi pencatatan penjualan. Kemudian, studi literatur melibatkan penelusuran jurnal ilmiah dan dokumentasi resmi tentang pengembangan PWA, khususnya dalam pengelolaan *cache*, serta mengenai teknik pengujian efisiensi *cache* terhadap kinerja aplikasi.

2.2 Analisis dan Perancangan Sistem

Untuk menganalisis dan merancang sistem, penelitian ini menggunakan metode *Rapid Application Development* (RAD). Metode RAD dipilih karena mampu mempercepat proses pengembangan aplikasi melalui iterasi yang cepat dan umpan balik pengguna secara kontinu [8]. Tahapan dalam metode RAD, meliputi:

2.2.1 *Requirements Planning* (Perencanaan Kebutuhan)

Tahapan ini dilakukan untuk memperoleh daftar kebutuhan fungsional dan non-fungsional yang harus dipenuhi oleh aplikasi yang dikembangkan. Selain itu, direncanakan juga kebutuhan data, seperti logo usaha, daftar produk yang dijual, dan catatan penjualan sebelumnya; serta kebutuhan sumber daya, seperti perangkat dan layanan *hosting* yang akan digunakan. Kebutuhan fungsionalitas aplikasi ditunjukkan melalui *use case diagram* pada Gambar 1.

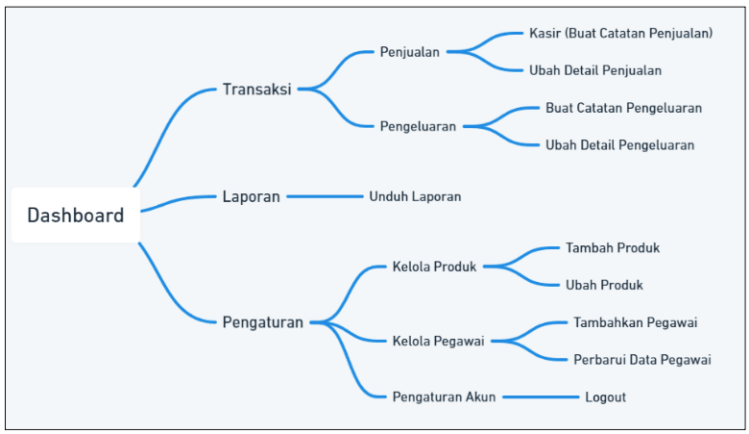


Gambar 1. *Use Case Diagram*

Gambar 1 merupakan *use case diagram* untuk aplikasi pencatatan penjualan yang dikembangkan. Berdasarkan diagram tersebut, aplikasi ini melibatkan dua aktor utama, yaitu pemilik usaha yang berperan sebagai admin dan pegawai sebagai pengguna biasa. Admin memiliki kemampuan untuk melakukan berbagai tugas seperti mengelola data pegawai, produk, dan catatan penjualan, serta melihat laporan transaksi dan mencatat pengeluaran. Sementara itu, pegawai memiliki akses hampir sama dengan admin, kecuali fitur pengelolaan data pegawai, pengelolaan produk, dan mengunduh laporan transaksi. Untuk mengakses seluruh fitur tersebut, pengguna diharuskan untuk melakukan *login* terlebih dahulu.

2.2.2 *User Design* (Desain Pengguna)

Pada tahap ini pengembang akan berkoordinasi pemilik usaha ayam geprek dalam merancang tampilan antarmuka yang mereka inginkan. Perancangan tampilan antarmuka aplikasi menggunakan pendekatan desain *mobile-first* yang mengutamakan pengalaman pengguna pada perangkat *mobile*, kemudian baru dilakukan penyesuaian tampilan pada desktopnya [9]. Tahapan ini menghasilkan sebuah *sitemap* dan prototipe aplikasi. Rancangan sitemap yang dapat dilihat pada Gambar 2.



Gambar 2. *Sitemap* Aplikasi

2.2.3 *Construction* (Konstruksi)

Tahapan selanjutnya dari model RAD adalah *construction*, yaitu proses pembangunan aplikasi berdasarkan prototipe desain sistem yang telah dirancang. Pada tahap ini, dilakukan penyusunan kode sumber dalam ekosistem *JavaScript* dengan kerangka kerja *Vue.js* pada sisi *frontend* dan penggunaan *NodeJS* dengan kerangka kerja *Express.js* pada sisi *backend*, serta basis data PostgreSQL.

2.2.4 *Cutover* (Pemindahan)

Setelah pengembangan aplikasi mencapai tingkat yang diinginkan, tahap *cutover* dilakukan. Tahap ini melibatkan pengujian fitur keseluruhan, pelatihan pengguna, dan pemindahan aplikasi dari lingkungan pengembangan (*development*) ke lingkungan produksi (*production*).

2.3 Konfigurasi Service Worker

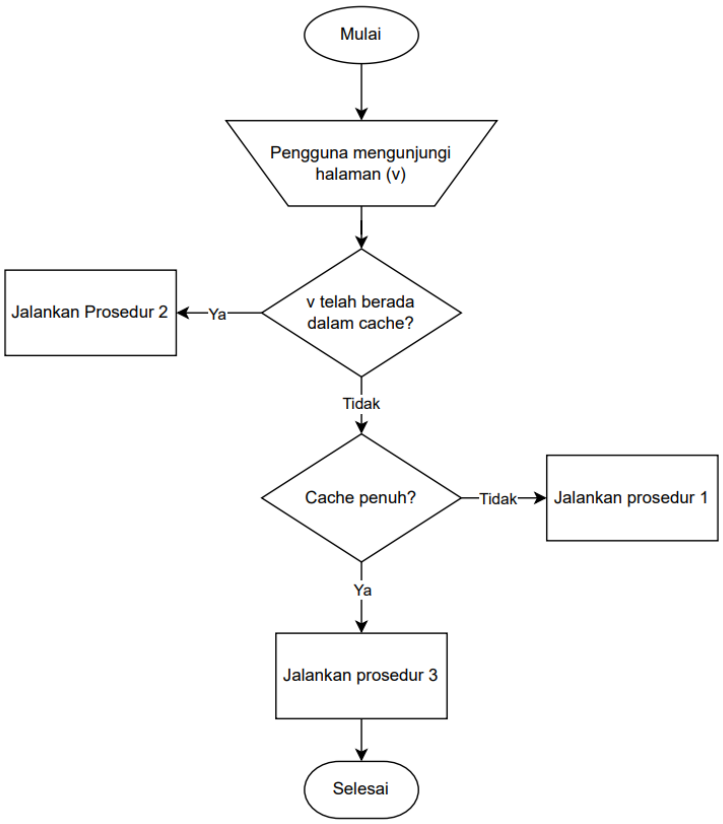
Konfigurasi *service worker* dilakukan setelah keseluruhan desain antarmuka telah diimplementasikan dan aplikasi telah diintegrasikan pada *backend* API. *Service Worker* memegang peran krusial dalam PWA dengan bertindak sebagai *proxy* di antara browser dan server [4]. Fungsinya mencakup menangani pengaksesan aplikasi secara *offline* dan meningkatkan kecepatan akses halaman dengan menyimpan berkas statis ke dalam *cache* dengan lebih mudah [10]. *Service worker* tersusun oleh baris kode *JavaScript* yang beroperasi di latar belakang untuk mengelola *cache*, menyinkronisasi data, dan berinteraksi dengan jaringan [11]. Berdasarkan siklus hidup *service worker*, terdapat 3 *event* utama dalam menangani *cache* yaitu *install event*, *activate event*, dan *fetch event* [12]. Dalam siklus hidupnya, *service worker* melakukan install sekali untuk menyimpan aset penting dalam *cache*, dilanjutkan proses *activate* untuk memperbarui dan membersihkan *cache* lama, serta menjalankan *fetch* untuk setiap permintaan HTTP untuk mendapatkan respons data, baik dari server, maupun dari *cache* jika tersedia.

2.4 *Fuzzy Page Replacement Algorithm* (FPRA) dalam Pengelolaan *Cache*

Konsep dasar dari *cache* adalah sebuah hit terjadi ketika sebuah halaman di dalam *cache* direferensikan (diakses). Sebaliknya, *miss* akan terjadi jika akses merujuk ke halaman yang tidak ada dalam *cache* sehingga halaman yang direferensikan harus ditambahkan. Namun, ketika *cache* penuh, halaman sebelumnya harus dihapus untuk memberi ruang pada halaman baru. Penentuan halaman inilah yang menjadi fokus implementasi FPRA. FPRA memilih halaman anggota dengan prioritas terendah dari klaster yang telah dibentuk menggunakan algoritma FCM yang memiliki jarak *euclidean* paling dekat dengan pusatnya untuk dihapus dari *cache* setiap kali terjadi *miss* sehingga halaman yang baru diakses bisa disimpan dalam *cache*.

2.4.1 Alur Kerja FPRA

Fungsi utama FPRA dijalankan setiap pengguna mengakses suatu halaman. Algoritma dari FPRA dapat dilihat pada diagram alur Gambar 3. Pengkondisian dilakukan dengan mengecek data yang tersimpan dalam *cache* dan kapasitas dari *cache*.



Gambar 3. Diagram Alur Fungsi Utama FPRA

Berdasarkan fungsi utama tersebut, kondisi *cache* terbagi menjadi 3, yaitu:
1) Kondisi pertama, yaitu ketika halaman yang baru diakses tidak ada dalam *cache* dan *cache* tidak penuh, maka prosedur 1 akan dijalankan. Prosedur 1 dapat dilihat pada Algoritma 1.

Algorithm 1 Prosedur 1 FPRA

```
Inisialisasi v sebagai halaman yang diakses
Inisialisasi v sebagai halaman yang diakses
Inisialisasi daftar halaman (dalam bentuk array)
Tambahkan v ke daftar halaman
for setiap slot di memori cache do
    if slot kosong then
        Simpan j ke dalam cache
        Inisialisasi tingkat kebaruan v = 1
        Inisialisasi frekuensi akses v = 1
        Inisialisasi tingkat referensi v = 1
        Jalankan algoritma FCM untuk memperbarui klaster
    end if
end for
```

2) Kondisi kedua, yaitu ketika halaman yang baru diakses telah berada dalam cache, maka prosedur 2 pada Algoritma 2 dijalankan.

Algorithm 2 Prosedur 2 FPRA

Inisialisasi v sebagai halaman yang sedang diakses / direferensikan
Inisialisasi w sebagai halaman lain dalam cache
for setiap w dalam cache **do**
    if $w \neq v$ **then**
        Tingkat kebaruan w bertambah 1
    end if
end for
Tingkat referensi v = Tingkat referensi v + (Tingkat kebaruan / 2)
Frekuensi akses v bertambah 1
Tingkat kebaruan v berkurang 1
Jalankan algoritma FCM untuk memperbarui klaster

3) Kondisi ketiga, yaitu ketika halaman yang baru diakses tidak ada dalam *cache* dan *cache* sudah penuh, maka prosedur 3 yang ditunjukkan pada Algoritma 3 akan dijalankan.

Algorithm 3 Prosedur 3 FPRA

Inisialisasi v sebagai halaman yang sedang diakses / direferensikan
Inisialisasi w sebagai halaman lain dalam cache
Tambahkan v ke dalam daftar halaman
Panggil Algoritma 4 FCM untuk memperbarui klaster
Tentukan klaster dengan nilai prioritas (CP) terkecil
Pilih halaman korban (t) yang memiliki jarak Euclidean terkecil untuk dihapus dari cache
Tempatkan v pada slot halaman korban
for setiap w dalam cache **do**
    if $w \neq v$ **dan** $w \neq t$ **then**
        Tingkat kebaruan w bertambah 1
    end if
end for
Inisialisasi tingkat referensi $v = 1$
Inisialisasi frekuensi akses $v = 1$
Inisialisasi tingkat kebaruan $v = 1$
Jalankan algoritma FCM untuk memperbarui klaster

2.4.2 Pengklasteran Halaman Menggunakan Fuzzy C-Means (FCM)

FCM bertujuan untuk meminimalkan fungsi dan memperoleh pusat klaster untuk mengidentifikasi data yang masuk ke dalam klaster tersebut [13]. Dalam hal ini, terdapat tiga parameter yang digunakan untuk perhitungan, yaitu tingkat kebaruan, frekuensi pengaksesan, dan tingkat referensi dari setiap data halaman. Dengan jumlah klaster adalah 3, tahapan klasterisasi halaman dengan FCM ditunjukkan pada Algoritma 4. Berdasarkan Algoritma 4, klaster dianggap konvergen ketika pusat klaster tidak berubah secara signifikan atau ketika perubahan matriks keanggotaan telah berada di bawah ambang batas atau telah mencapai iterasi maksimum.

Algorithm 4 Klasterisasi Halaman dengan FCM

Inisialisasi data (x), jumlah klaster (c), nilai ambang batas minimum error / epsilon (e), dan jumlah maksimum iterasi
Inisialisasi derajat keanggotaan (U) setiap data pada masing-masing klaster
while iterasi kurang dari jumlah maksimum iterasi **do**
    Hitung centroid (titik pusat) setiap klaster
    Perbarui derajat keanggotaan baru
    Hitung selisih antara centroid dengan titik data (jarak Euclidean)
    Hitung nilai jarak relatif antara titik data dari centroid satu dengan centroid lainnya
    if perubahan jarak berada di bawah batas minimum error **then**
        Hentikan perulangan **break**
    end if
    Gunakan derajat keanggotaan baru untuk perhitungan centroid
    Iterasi bertambah
end while
Ulangi langkah 3 hingga klaster mencapai kondisi konvergen

2.4.3 Penentuan Klaster Prioritas (CP)

Untuk menentukan halaman mana yang akan dihapus dan mana yang tetap dipertahankan dalam *cache*, perlu dihitung nilai CP untuk masing-masing klaster. Halaman anggota pada klaster dengan nilai CP terendah akan dihapus. Nilai CP dapat dihitung menggunakan Persamaan (1).

$$CP_i = \frac{CF_i}{CR_i} \times C\delta_{tk}(i) \tag{1}$$

CP_i : Nilai prioritas klaster i
CF_i : Nilai rata-rata frekuensi akses halaman dalam klaster i
CR_i : Nilai rata-rata tingkat kebaruan halaman dalam klaster i
$C\delta_{tk}(i)$: Nilai rata-rata tingkat kebaruan halaman dalam klaster i

2.5 Pengujian dan Evaluasi Kinerja Aplikasi

Pada tahapan pengujian, penelitian ini akan mengkomparasikan hasil pengujian kinerja aplikasi ketika sebelum menerapkan FPRA dengan setelah menerapkan FPRA dalam pengelolaan *cache*. Pengujian dilakukan dalam tiga jenis pengujian, yaitu menggunakan alat berupa *Lighthouse*, pengujian efektivitas FPRA terhadap *cache hit ratio*, serta pengujian manual terhadap waktu muat halaman.

2.5.1 Pengujian Otomatis Menggunakan *Lighthouse*

Pada pengujian pertama, kinerja aplikasi diuji menggunakan *Lighthouse*, yaitu alat pengujian gratis yang telah disertakan dalam *Google Chrome Developer Tools* dan dapat digunakan untuk menguji kinerja aplikasi *mobile* dan *desktop* dengan mengumpulkan berbagai metrik aplikasi secara otomatis [14]. Metrik yang akan diujikan menggunakan *Lighthouse* antara lain: *Performance Score*, *First Contentful Paint* (FCP), *Largest Contentful Paint* (LCP), *Cumulative Layout Shift* (CLS), T*otal Blocking Time* (TBT), *Speed Index* (SI), *Accessibility Score*, dan *Best Practices Score*. Setiap metrik memiliki nilai yang dianggap baik berdasarkan standar yang telah ditetapkan, misalnya FCP di bawah 3 detik, LCP di bawah 4 detik, CLS kurang dari 0.25, TBT tidak lebih dari 600 ms, dan SI tidak lebih dari 5.5 detik. Hasil pengujian ditampilkan dalam skor dan warna yang memberikan interpretasi tentang kualitas performa situs web [15]. Untuk keterangan warna dan rentang skor kinerja dalam hasil pengujiannya ditunjukkan pada Tabel 1.

Tabel 1. Kode Warna dan Rentang Skor Kinerja [15]

Rentang Skor	Warna	Keterangan
0 – 49	Merah	Buruk
50 – 89	Oranye	Perlu peningkatan (sedang)
90 – 100	Hijau	Baik

2.5.2 Pengujian Efektivitas FPRA Terhadap Cache Hit Ratio

Pengujian ini bertujuan untuk mengukur hit dan *miss ratio* untuk mendapatkan gambaran langsung tentang seberapa efektif performa algoritma pengelolaan *cache* bekerja dalam skenario akses tertentu. Semakin tinggi *hit ratio*, semakin efisien *cache* dalam mengurangi waktu akses ke server [7]. Pada pengujian ini, algoritma FPRA akan dibandingkan dengan algoritma *Least Frequently Used* (LFU) dan algoritma tradisional *First-In-First-Out* (FIFO). Algoritma LFU mempertimbangkan tingkat frekuensi halaman untuk menentukan item mana yang akan dihapus dari *cache* saat kapasitasnya penuh, dengan item dengan frekuensi terendah dihapus terlebih dahulu. Sedangkan algoritma FIFO menggunakan pendekatan dasar dengan selalu mengeluarkan objek tertua dari *cache* [16]. Ketiga algoritma tersebut akan diterapkan secara bergantian pada sistem. Selanjutnya, total hit dan *miss* akan dicatat dari 50 permintaan atau pengaksesan halaman secara acak, dengan urutan yang sama pada ketiga algoritma.

2.5.3 Pengujian Waktu Muat Halaman dalam Berbagai Koneksi Jaringan

Pengujian ini dilakukan secara manual menggunakan beberapa skenario untuk menguji waktu muat halaman di berbagai kondisi jaringan untuk masing-masing algoritma FPRA, LFU, dan FIFO. Hal ini berguna

untuk mengetahui lebih detail bagaimana penerapan *cache* dapat berpengaruh terhadap waktu muat halaman. Skenario pengujian yang digunakan ditunjukkan pada Tabel 2.

Tabel 2. Skenario Pengujian Waktu Muat Halaman

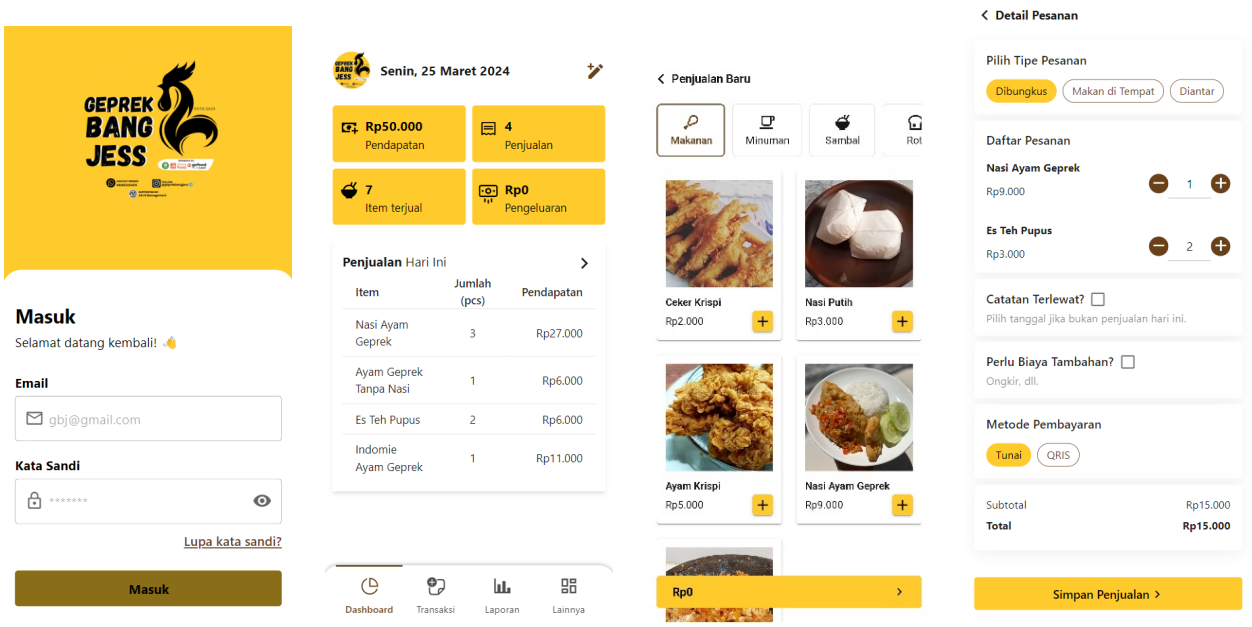
Skenario	Kunjungan ke-	Service Worker	Cache	Koneksi Jaringan
1	1	Tidak aktif	Tidak tersedia	Wi-Fi (5G)
2	>1	Aktif	Tersedia	Wi-Fi (5G)
3	1	Tidak Aktif	Tidak tersedia	4G
4	>1	Aktif	Tersedia	4G
5	1	Tidak Aktif	Tidak Tersedia	3G
6	>1	Aktif	Tersedia	3G
7	>1	Aktif	Tersedia	Offline

3. Hasil

Pada bagian ini akan membahas hasil dari implementasi metode penelitian yang dijelaskan sebelumnya, meliputi hasil implementasi antarmuka, web *manifest* untuk PWA, implementasi FPRA dalam *cache*, dan hasil pengujian aplikasi.

3.1 Tampilan Antarmuka Aplikasi

Tampilan antarmuka aplikasi dibangun berdasarkan desain yang telah dirancang sebelumnya, dengan menyesuaikan preferensi pengguna dan mengikuti struktur *site map* yang telah disusun. Aplikasi yang dikembangkan pada penelitian ini memiliki total halaman sebanyak 24 halaman, termasuk halaman utama setiap fitur dan halaman detailnya. Berapa contoh tampilannya ditunjukkan pada Gambar 4.



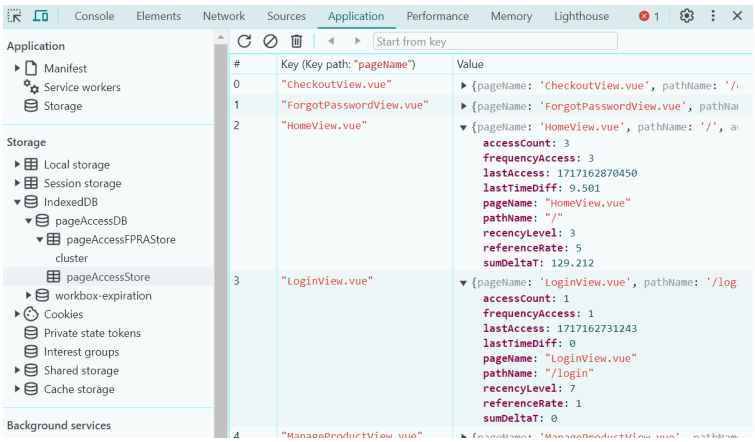
(a) *Login* **(b)** *Dashboard* **(c)** Penjualan Baru **(d)** Tambah Catatan Penjualan

Gambar 4. Tampilan Halaman Antarmuka Aplikasi

3.2 Implementasi *Fuzzy Page Replacement Algorithm* (FPRA)

Dalam penelitian ini, FPRA digunakan untuk mengelola *cache* halaman aplikasi menggunakan tiga parameter, yaitu tingkat kebaruan, tingkat frekuensi akses, dan tingkat referensi setiap halaman. Parameter

tersebut akan diproses menggunakan algoritma FCM untuk membentuk klaster yang memuat halaman-halaman yang telah disimpan dalam *cache*. Untuk menyimpan nilai parameter tersebut, aplikasi ini menggunakan *IndexedDB* sebagai *database* lokal pada sisi *client* [17]. Contoh penyimpanan informasi halaman yang digunakan sebagai parameter FPRA dalam *IndexedDB* ditunjukkan pada Gambar 5.

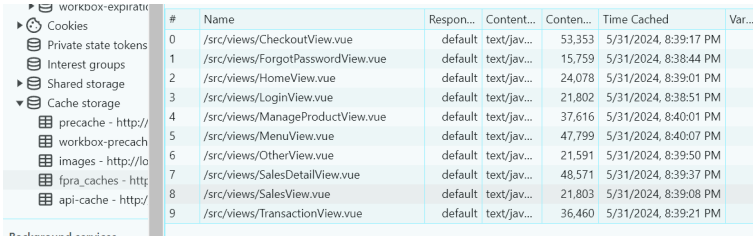


Gambar 5. Penyimpanan Informasi Halaman dan Parameter FRPA dalam *IndexedDB*

Cara kerja dari algoritma yaitu dengan memperbarui informasi akses halaman yang sedang dibuka di dalam *IndexedDB* setiap kali pengguna membuka suatu halaman. Selanjutnya, aplikasi akan mengirimkan pesan kepada *service worker* yang menginformasikan bahwa pengaksesan halaman telah terjadi dan fungsi utama FPRA perlu dijalankan. Berikut ini merupakan contoh hasil implementasi klasterisasi halaman menggunakan algoritma FCM, ditunjukkan pada Gambar 6.

Gambar 6. Hasil Klasterisasi Halaman Menggunakan FCM

Anggota klaster dibatasi 10 halaman sehingga ketika cache penuh seperti yang ditunjukkan pada Gambar 7, algoritma akan mencari klaster dengan nilai prioritas (CP) terkecil dan menentukan halaman mana yang akan dihapus dari *cache*. Nilai prioritas setiap klaster dihitung dengan Persamaan (1).



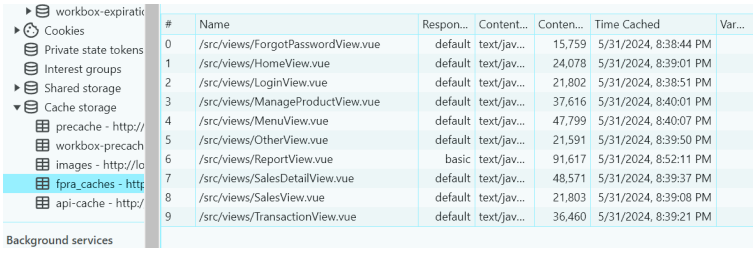
Gambar 7. Data dalam *Cache* Sebelum Pergantian Halaman

Anggaplah pengguna baru saja mengakses halaman laporan (*ReportView*) yang tidak ada dalam *cache* dan *cache* sudah penuh. Setelah dilakukan perhitungan seperti pada Gambar 8, maka didapatkan bahwa klaster dengan nilai prioritas terendah adalah klaster 2 dengan nilai CP = 0,1470 dan halaman yang akan dihapus oleh FPRA adalah halaman *checkout*.

```
Anggota cluster: ▶ (11) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]   dev-sw.js:531
Silhouette Score: 0.4974008778493655                                                dev-sw.js:535
data parameter ▶ (11) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]     dev-sw.js:770
Anggota Cluster 1:                                                                  dev-sw.js:876
▶ (5) ['ManageProductView.vue', 'OtherView.vue', 'ReportView.vue', 'SalesView.vue', 'TransactionView.vue']
CP_1 : 0.9                                                                          dev-sw.js:900
Anggota Cluster 2:                                                                  dev-sw.js:876
▶ (5) ['CheckoutView.vue', 'ForgotPasswordView.vue', 'LoginView.vue', 'MenuView.vue', 'SalesDetailView.vue']
CP_2 : 0.14705882352941177                                                          dev-sw.js:900
Anggota Cluster 3: ▶ ['HomeView.vue']                                               dev-sw.js:876
CP_3 : 5                                                                            dev-sw.js:900
Klaster dengan prioritas terendah: cluster 2                                        dev-sw.js:772
Halaman korban: CheckoutView.vue                                                    dev-sw.js:773
Halaman 'CheckoutView.vue' telah dihapus dari store 1.                              dev-sw.js:945
```

Gambar 8. Penentuan Klaster Prioritas dan Halaman Korban

Setelah halaman korban dikeluarkan, halaman baru akan menggantikan posisinya dalam *cache* (lihat dan dilanjutkan dengan proses klasterisasi kembali setelah pengguna mengunjungi halaman baru).

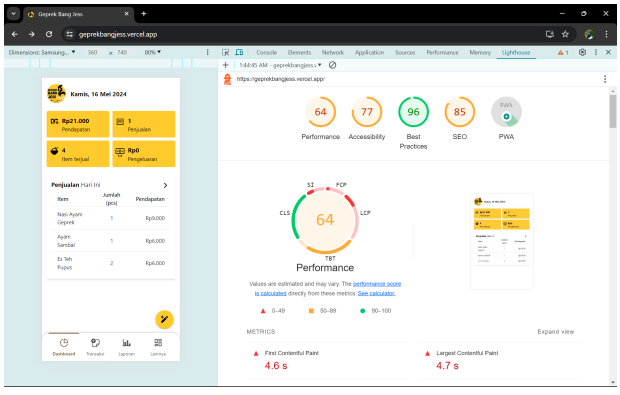


Gambar 9. Data dalam *Cache* Setelah Pergantian Halaman dengan FPRA

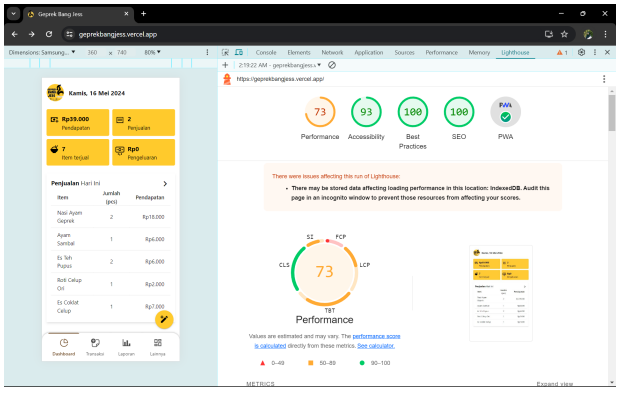
3.3 Hasil Pengujian Aplikasi

3.3.1 Pengujian Menggunakan *Lighthouse*

Pengujian dilakukan pada dua kondisi, yaitu saat sebelum FPRA diimplementasikan dan setelah FPRA diimplementasikan.



(a) Sebelum Implementasi FPRA



(b) Setelah Implementasi FPRA

Gambar 10. Hasil Pengujian Performa Aplikasi Menggunakan *Lighthouse*

Gambar 10 (a) merupakan hasil pengujian dengan *Lighthouse* untuk kondisi sebelum menerapkan FPRA, tepatnya ketika aplikasi telah menggunakan konfigurasi bawaan dari *service worker*. Hasil pengujian selengkapnya ditunjukkan pada Tabel 3.

Tabel 3. Hasil Pengujian *Lighthouse* Sebelum Implementasi FPRA

Metrik	Hasil	
	Tanpa *Cache*	**Dengan *Cache***
Performance	48/100	64/100
First Contentful Paint	4.6 s	4.6 s
Largest Contentful Paint	5.0 s	4.7 s
Speed Index	5.8 s	6.0 s
Total Blocking Time	870 s	260 s
Cumulative Layout Shift	0.021	0.021
Accessibility	77/100	77/100
Best Practices	96/100	96/100
SEO	85/100	85/100

Berdasarkan hasil pengujian tersebut, dapat diketahui bahwa skor penilaian untuk metrik *Perfomance* meningkat sebanyak 16 poin, dari angka 48 menjadi 64 setelah menggunakan cache. Peningkatan secara drastis juga terjadi pada metrik Total *Blocking Time* dengan lebih cepat 610 ms ketika *cache* digunakan.

Tabel 4. Hasil Pengujian *Lighthouse* Setelah Implementasi FPRA

Metrik	Hasil	
	Tanpa *Cache*	**Dengan *Cache***
Performance	69/100	73/100
First Contentful Paint	3.9 s	3.8 s
Largest Contentful Paint	4.1 s	4.0 s
Speed Index	4.9 s	4.1 s
Total Blocking Time	300 s	250 s
Cumulative Layout Shift	0.01	0.019
Accessibility	93100	93/100
Best Practices	96/100	100/100
SEO	100/100	100/100

Sebaliknya, Gambar 10(b) merupakan hasil pengujian untuk kondisi setelah aplikasi menerapkan FPRA. Hasil pengujian selengkapnya ditunjukkan pada Tabel 4. Berdasarkan hasilnya, terjadi peningkatan skor pada hampir semua metrik yang diujikan setelah menggunakan *cache*. Performa aplikasi juga meningkat dari angka 69 ke 73, metrik *speed index* juga terlihat terjadi peningkatan yang cukup tinggi setelah *cache* digunakan.

3.3.2 Pengujian Efektivitas FPRA Terhadap *Cache Hit Ratio*

Pengujian dilakukan dengan mengunjungi 50 halaman secara acak pada aplikasi dengan menerapkan algoritma FIFO, LFU, dan FPRA untuk mengelola cache secara bergantian. Tercatat kondisi *hit* dan *miss* untuk setiap permintaan atau kunjungan yaitu pada Tabekl 5.

Tabel 5. Hasil Pengujian Hit/Miss Cache dengan Algoritma FIFO, LFU, dan FPRA

Urutan	Halaman	FIFO	LFU	FPRA	Urutan	Halaman	FIFO	LFU	FPRA
1	Login	Miss	Miss	Miss	**26**	Pengaturan	Hit	Hit	Hit
2	Beranda	Miss	Miss	Miss	**27**	Kelola Pegawai	Miss	Miss	Miss
3	Daftar Transaksi	Miss	Miss	Miss	**28**	Detail Pegawai	Miss	Miss	Miss
4	Tambah Penjualan	Miss	Miss	Miss	**29**	Kelola Pegawai	Hit	Miss	Hit
5	Checkout	Miss	Miss	Miss	**30**	Pengaturan	Hit	Hit	Hit
6	Daftar Transaksi	Hit	Hit	Hit	**31**	Profil	Miss	Miss	Miss
7	Detail Penjualan	Miss	Miss	Miss	**32**	Pengaturan	Hit	Hit	Hit
8	Beranda	Hit	Hit	Hit	**33**	Login	Miss	Miss	Miss
9	Tambah Penjualan	Hit	Hit	Hit	**34**	Beranda	Miss	Hit	Hit
10	Pengaturan	Miss	Miss	Miss	**35**	Daftar Transaksi	Miss	Hit	Hit
11	Kelola Produk	Miss	Miss	Miss	**36**	Tambah Pengeluaran	Miss	Hit	Miss
12	Edit Produk	Miss	Miss	Miss	**37**	Daftar Transaksi	Hit	Hit	Hit
13	Kelola Produk	Hit	Hit	Hit	**38**	Detail Pengeluaran	Hit	Hit	Hit
14	Beranda	Hit	Hit	Hit	**39**	Laporan	Hit	Hit	Miss
15	Tambah Penjualan	Hit	Hit	Hit	**40**	Laporan Pengeluaran	Miss	Miss	Miss
16	Checkout	Hit	Hit	Hit	**41**	Transaksi Penjualan	Hit	Hit	Hit
17	Pengaturan	Hit	Hit	Hit	**42**	Detail Penjualan	Miss	Miss	Miss
18	Kelola Produk	Hit	Hit	Hit	**43**	Pengaturan	Miss	Hit	Hit
19	Edit Produk	Hit	Hit	Hit	**44**	Kategori Produk	Miss	Miss	Miss
20	Kelola Produk	Hit	Hit	Hit	**45**	Kategori Pengeluaran	Miss	Miss	Miss
21	Daftar Transaksi	Hit	Hit	Hit	**46**	Kategori Produk	Hit	Miss	Hit
22	Tambah Penjualan	Hit	Hit	Hit	**47**	Kategori Pengeluaran	Hit	Miss	Hit
23	Checkout	Hit	Hit	Hit	**48**	Beranda	Hit	Hit	Hit
24	Daftar Transaksi	Hit	Hit	Hit	**49**	Daftar Transaksi	Hit	Hit	Hit
25	Laporan	Miss	Miss	Miss	**50**	Detail Penjualan	Hit	Miss	Miss

Berdasarkan hasil pengujian pada Tabel 5, didapatkan bahwa total *hit cache* yaitu: sebanyak 28 kali ketika menggunakan algoritma FIFO, 28 kali ketika menggunakan algoritma LFU, dan 29 kali ketika menggunakan FPRA. Sedangkan untuk total miss cache-nya, yaitu: 22 kali miss ketika menggunakan algoritma FIFO, 22 kali miss ketika menggunakan algoritma LFU, dan 29 kali miss ketika menggunakan FPRA. Dengan demikian, *hit ratio* dapat dihitung dengan Persamaan (2) dan *miss ratio* dihitung dengan Persamaan (3).

$$\text{Hit Ratio} = \frac{\text{total hit}}{\text{total seluruh permintaan}} \times 100\% \tag{2}$$

$$\text{Miss Ratio} = \frac{\text{total miss}}{\text{total seluruh permintaan}} \times 100\% \tag{3}$$

Setelah dilakukan perhitungan, maka didapatkan *hit ratio* dan *miss ratio* seperti pada Tabel 6.

Tabel 6. Hasil *Hit Ratio* dan *Miss Ratio*

Algoritma	Total Hit	Total Miss	Hit Ratio	Miss Ratio
FIFO	28	22	56.00%	44.00%
LFU	28	22	56.00%	44.00%
FPRA	29	21	58.00%	42.00%

3.3.3 Pengujian Waktu Muat Halaman Secara Manual

Jenis pengujian selanjutnya yang dilakukan pada penelitian ini adalah pengujian terhadap waktu muat halaman. Pengujian ini dilakukan dengan skenario pada Tabel 2. Hasil pengujiannya dapat dilihat pada

Tabel 7. Rata-rata waktu muat halaman ketika FPRA diterapkan adalah selama 660.14 milisecond, lebih cepat 178 milisecond daripada saat menerapkan FIFO dan LFU, dengan rata-rata waktu muat 838.17 milisecond. Grafik perbandingan waktu muat dapat dilihat pada Gambar 11.

4. Pembahasan

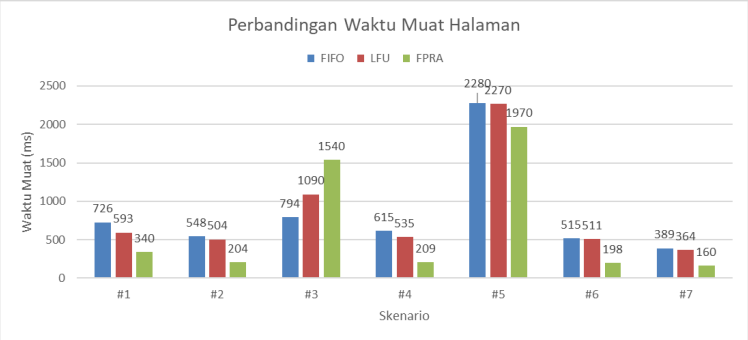
Implementasi pendekatan *fuzzy* dalam pengelolaan *cache* yang dimaksud dalam penelitian ini adalah dengan menggunakan algoritma pergantian halaman yang bernama *Fuzzy Page Replacement Algorithm* (FPRA). Dengan menggunakan model pengembangan *Rapid Application Development*, penelitian ini berhasil mengembangkan sebuah aplikasi pencatatan penjualan berbasis *progressive web app* yang sesuai dengan referensi pengguna dari studi kasus yang digunakan.

Berdasarkan pengujian efektivitas dari ketiga algoritma pergantian halaman yang diimplementasikan dalam pengelolaan *cache* aplikasi, terjadi peningkatan sebesar 2% pada *cache hit ratio* saat mengimplementasikan FPRA dibandingkan dengan algoritma FIFO dan LFU. Ini berarti bahwa lebih banyak permintaan data yang dapat dipenuhi dari *cache*, yang secara langsung mengurangi waktu muat halaman dan meningkatkan responsivitas aplikasi.

Tabel 7. Hasil Pengujian Waktu Muat Halaman

Skenario	Waktu Muat Halaman (ms)		
	FIFO	LFU	FPRA
1	726	593	340
2	548	504	204
3	794	1090	1540
4	615	535	209
5	2280	2270	1970
6	515	511	198
7	389	364	160
Rata-rata	**838.14**	**838.14**	**660.14**

Berdasarkan grafik pada Gambar 11, waktu muat mengalami penurunan yang signifikan saat ketiga algoritma diimplementasikan dalam berbagai kondisi jaringan ketika aplikasi menggunakan *cache* (skenario 2, 4, 6, 7), dibandingkan dengan kondisi tanpa penggunaan *cache* (skenario 1, 3, 5).



Gambar 11. Grafik Perbandingan Waktu Muat Halaman

Pembahasan untuk dampak penggunaan *cache* terhadap berbagai kondisi jaringan adalah sebagai berikut.

1) Koneksi Wi-Fi (5G)
   Pengujian pada skenario 1 dan 2 menunjukkan bahwa penggunaan *cache* menurunkan waktu muat halaman dari 726 ms ke 548 ms pada FIFO, dari 593 ms ke 504 ms pada LFU, dan dari 340 ms ke 204

ms pada FPRA. Hal ini berarti penggunaan *cache* secara signifikan meningkatkan efisiensi waktu muat, dengan FPRA menunjukkan performa terbaik.

2) Koneksi 4G
   Pengujian pada skenario 3 dan 4 menunjukkan bahwa penggunaan *cache* menurunkan waktu muat halaman dari 794 ms ke 615 ms pada FIFO, dari 1090 ms ke 535 ms pada LFU, dan dari 1540 ms ke 209 ms pada FPRA. Hal ini berarti *cache* secara substansial memperbaiki waktu muat, dengan FPRA memberikan hasil yang paling optimal.
3) Koneksi 3G
   Pengujian pada skenario 5 dan 6 menunjukkan bahwa penggunaan *cache* menurunkan waktu muat halaman dari 2280 ms ke 515 ms pada FIFO, dari 2270 ms ke 511 ms pada LFU, dan dari 1970 ms ke 198 ms pada FPRA. Hal ini berarti *cache* memberikan peningkatan performa yang signifikan pada jaringan 3G, dengan FPRA menunjukkan pengurangan waktu muat yang paling besar.
4) Tanpa Internet (*Offline*)
   Pengujian pada skenario 7 menunjukkan waktu muat halaman 389 ms pada FIFO, 364 ms pada LFU, dan 160 ms pada FPRA. Hal ini berarti bahwa dalam kondisi *offline*, penggunaan *cache* dengan FPRA menghasilkan waktu muat yang paling cepat dibandingkan algoritma lainnya.

Secara keseluruhan, implementasi *cache* terbukti efektif dalam mengurangi waktu muat halaman pada berbagai kondisi jaringan, dengan algoritma FPRA secara konsisten memberikan hasil terbaik dalam pengurangan waktu muat sebesar 21% berdasarkan rata-ratanya.

5. Simpulan

Pendekatan *fuzzy* yang diterapkan melalui implementasi *Fuzzy Page Replacement Agorithm* (FPRA) dalam pengelolaan *cache* pada aplikasi pencatatan penjualan yang dikembangkan dalam penelitian ini dapat meningkatkan waktu muat halaman hingga 21% lebih cepat. Penerapan FPRA pada *service worker* juga dapat meningkatkan rata-rata skor performa aplikasi sebesar 15 poin dari angka 56 menjadi 71, yang berarti aplikasi ini telah berstatus cukup baik, tetapi perlu ditingkatkan, dari segi kinerjanya. Dengan mempertahankan data yang lebih relevan dalam *cache*, FPRA mendapatkan nilai *cache hit ratio* tertinggi dibandingkan dengan algoritma FIFO dan LFU. Hal ini disebabkan karena FPRA lebih adaptif dalam pengambilan keputusan untuk menggantikan halaman *cache* dengan mempertimbangkan tiga parameter.

Pustaka

[1] T. D. R. Sari, D. T. Kencana, and M. Anjelita, "Pelatihan penggunaan aplikasi penjualan," *J. Soc. Sci. Technol. Community Serv.*, vol. 4, no. 1, pp. 126–142, 2023.

[2] V. S. Magomadov, "Exploring the role of progressive web applications in modern web development," in *Journal of Physics: Conference Series*, 2020.

[3] H. T. Ramdani, N. Ainun, and A. M. B, "Implementation of progressive web app on dropship data management application to anticipate product order errors," *J. Inf. Syst. Technol. Eng.*, vol. 1, no. 2, pp. 38–42, Jun 2023.

[4] R. Fauzan, I. Krisnahati, B. D. Nurwibowo, and D. A. Wibowo, "A systematic literature review on progressive web application practice and challenges," *IPTEK J. Technol. Sci.*, vol. 33, no. 1, p. 43, 2022.

[5] W. Chao, "Web cache intelligent replacement strategy combined with gdsf and svm network re-accessed probability prediction," *J. Ambient Intell. Humaniz. Comput.*, vol. 11, no. 2, pp. 581–587, 2020.

[6] A. Adha, "Penerapan logika fuzzy pada mesin cuci dan menentukan lama waktu pencucian," *JIKO (Jurnal Inform. dan Komputer)*, vol. 6, no. 1, p. 125, Feb 2022.

[7] D. A. Bengar, A. Ebrahimnejad, H. Motameni, and M. Golsorkhtabaramiri, "A page replacement algorithm based on a fuzzy approach to improve cache memory performance," *Soft Comput.*, vol. 24, no. 2, pp. 955–963, 2020.

[8] A. P. Sari, M. Muharrom, A. Haromainy, and R. Purnomo, "Implementasi metode rapid application development pada aplikasi sistem informasi monitoring santri berbasis website," *Decod. J. Pendidik. Teknol. Inf.*, vol. 4, no. 1, pp. 316–325, 2024.

[9] Zulkarnain, "Penerapan mobile-first design pada antarmuka website profil sekolah menggunakan metode human-centred design (studi kasus: Smpn 21 malang)," *J. Ilm. Teknol. Inf. Asia*, vol. 13, no. 2, pp. 125–136, 2019.

[10] A. A. Kurniawan, "Analisis performa progressive web application (pwa) pada perangkat mobile," *J. Ilm. Inform. Komput.*, vol. 25, no. 1, pp. 18–31, 2020.

[11] S. Aripin and Somantri, "Implementasi progressive web apps (pwa) pada repository e-portofolio mahasiswa," *J. Eksplora Inform.*, vol. 10, no. 2, pp. 148–158, 2021.

[12] A. Gambhir and G. Raj, "Analysis of cache in service worker and performance scoring of progressive web application," in *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)*, 2018, pp. 294–299.

[13] M. Y. R., L. N. Hayati, and Sugiarti, "Sistem pendukung keputusan penilaian kinerja pegawai dengan algoritma fuzzy c-means dan hungarian," *JIKO (Jurnal Inform. dan Komputer)*, vol. 7, no. 2, pp. 229–243, 2023.

[14] F. Correia, O. Ribeiro, and J. C. Silva, "Progressive web apps development: Study of caching mechanisms," in *2021 21st International Conference on Computational Science and Its Applications, ICCSA 2021*, 2021, pp. 181–187.

[15] C. for Developer, "Lighthouse performance scoring," accessed May 15, 2024. [Online]. Available: https://developer.chrome.com/docs/lighthouse/performance/performance-scoring

[16] D. Meint and S. Liebald, "From fifo to predictive cache replacement," in *Seminar Innovative Internet Technologies and Mobile Communications (IITM)*, 2019, pp. 25–31.

[17] Z. Tahir, A. A. Ilham, M. Niswar, and A. A. Fauzy, "Progressive web apps development and analysis with angular framework and service worker for e-commerce system," in *2021 IEEE Int. Conf. Comput.*, 2021, pp. 192–195.