Platform E-Learning untuk Pembelajaran Pemrograman Web Menggunakan Konsep Progressive Web Apps

Laurensius Adi, Rizky Januar Akbar, Wijayanti Nurul Khotimah
Departemen Teknik Informatika, Fakultas Informasi dan Teknologi Informatika Institut Teknologi Sepuluh Nopember (ITS)
e-mail: rizky@if.its.ac.id

***Abstrak*—Platform e-Learning bisa menjadi alat bantu proses belajar yang efektif, karena peserta didik bisa belajar dengan sendiri dan dari tempat yang tidak terbatas, selama masih ada koneksi internet. Yang sering menjadi kendala adalah ketersediaan platform e-Learning yang stabil dalam koneksi internet yang minim atau kondisi offline. Progressive Web Apps (PWA) adalah konsep pengembangan aplikasi berbasis web yang mencakup penerapan teknologi terbaru dari browser seperti service workers dan app manifest. Konsep PWA dapat memberikan pengalaman terbaik dalam menggunakan suatu aplikasi web walaupun dalam koneksi internet yang minim atau offline sekalipun dengan menggunakan service worker. Service worker sebagai pengatur request dan response pada sebuah aplikasi web dapat dirancang sesuai kebutuhan. Pada studi ini, pada service worker digunakan strategi caching network first, cache fallback dengan tambahan cache only pada berkas-berkas statis. Strategi ini dipilih untuk menghindari eror token pada kerangka kerja Laravel dan tetap mendapat respons yang cepat ketika sebuah halaman web dibuka. Penerapan konsep PWA khususnya service worker meningkatkan performa platform e-Learning terutama waktu memuat halaman menjadi lebih cepat dan dapat berjalan secara offline.**

***Kata Kunci—e-Learning, Progressive Web Apps, offline, MochaJS*.**

I. PENDAHULUAN

E-LEARNING sebagai salah satu metode pembelajaran di perkuliahan yang sangat potensial. Tak jarang banyak instansi pendidikan yang menggunakan platform *e-Learning* sebagai media belajar para mahasiswanya. Apabila dipergunakan dengan sepenuhnya, *e-Learning* bisa menjadi alat bantu proses belajar yang efektif, karena peserta didik bisa belajar dengan sendiri dan dari tempat yang tidak terbatas, selama masih ada koneksi internet. Yang sering menjadi kendala adalah ketersediaan platform e-Learning yang memenuhi kebutuhan dan stabil walaupun dalam koneksi internet yang minim atau kondisi *offline* sekalipun.

Progressive Web Apps (PWA) adalah penamaan untuk konsep baru yang dikemukakan oleh Alex Russell dan Frances Berriman pada tahun 2015 [1]. Konsep ini mencakup penerapan teknologi baru dari web *browser* seperti *service workers* dan app manifest. PWA memiliki karakteristik utama dapat diandalkan (*reliable*), cepat (*fast*), dan menarik (*engaging*), memastikan pengguna mendapatkan pengalaman terbaik dalam menggunakan suatu aplikasi web walaupun dalam koneksi internet yang minim atau *offline* sekalipun.

Dalam studi yang penulis ajukan ini, akan dibuat platform *e-Learning* yang dapat membantu proses pembelajaran di Teknik Infromatika ITS mata kuliah Pemrograman Web, dengan *preview* hasil pekerjaan dan bantuan pengecekan secara instan, serta mampu berjalan walaupun dalam kondisi *offline* menggunakan *service worker* yang diimplementasikan melalui konsep PWA.

II. TINJAUAN PUSTAKA

A. Progressive Web Apps

Progressive Web Apps (PWA) adalah konsep pengalaman pengguna yang mengabungkan bagian terbaik web dan bagian terbaik *native apps*. PWA berguna bagi pengguna sejak pertama membuka halaman sebuah web dengan konsep PWA, dan seiring dengan pengguna menggunakan aplikasi web lebih banyak lagi, aplikasi akan menjadi semakin *powerful*. Aplikasi dapat dimuat dengan cepat, bahkan dalam kondisi internet yang kurang baik, bisa mengirim *push notifications*, punya ikon aplikasi di *home screen*, dan bisa berjalan dalam mode layar penuh [1].

PWA sepenuhnya mengandalkan *browser* pengguna dan teknologi yang ada didalamnya. Sampai saat studi ini ditulis, sudah ada 73,61% dari seluruh *browser* di seluruh dunia yang mendukung fitur *service worker*, seperti Mozilla Firefox, Google Chrome, Chrome for Android dan Opera, sementara Edge dan Safari belum mendukung fitur ini [2].

B. Cache

Cache interface menyediakan mekanisme penyimpanan untuk pasangan objek *Request* dan *Response* mau disimpan ke dalam *cache*, contohnya sebagai bagian dari daur hidup *service worker*. Perlu diketahui bahwa *cache interface* terbuka terhadap halaman web dan juga *workers*. *Cache* tidak harus selalu digunakan bersamaan dengan *service worker* walaupun *cache* tercantum di dalam spesifikasinya [3]. *Cache* digambarkan sebagai sebuah *array* berisi objek *Request* yang bertindak sebagai pasangan untuk responsnya yang disimpan di dalam *browser* [4].

C. Service Worker

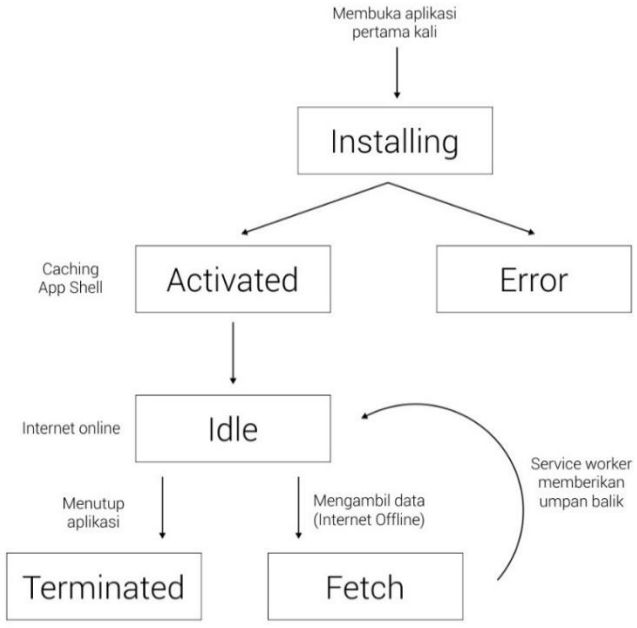
Service worker adalah salah satu jenis dari *web worker*, yaitu

script yang berjalan di belakang *browser* pengguna. *Service worker* pada dasarnya adalah berkas JavaScript yang berjalan pada *thread* yang berbeda dengan *main thread browser*, menangani *network request*, *caching*, mengembali-kan *resource* dari *cache*, dan bisa mengirimkan *push message*.

Aset web dapat disimpan sebagai *cache* lokal, sehingga dengan jaringan internet yang kurang memadai pun, pengguna tetap mendapat pengalaman yang baik. Aplikasi dapat tetap menjalankan halaman web yang sudah di-*cache* atau memberikan status koneksi tanpa *browser* menampilkan tulisan eror karena ketiadaan koneksi internet.

Untuk memasang *service worker* ke laman, kita butuh meregistrasinya dengan menggunakan JavaScript yang ada di halaman web. Setelah diregistrasi, *browser* akan memulai tahap *install service worker* di latar. Setelah aktif, *service worker* akan menangani semua halaman di bawah *scope* dimana *service worker* di-*install*, lalu akan ada 2 kemungkinan *state*: *terminated* untuk menghemat memori, atau menangani *fetch* ketika ada *network request* dari halaman web.

Service worker bisa menangani berbagai jenis *request*, tetapi yang bisa disimpan kedalam *cache* hanya semua *request* jenis GET. Pada *service worker*, bisa aplikasikan strategi *caching* sesuai keinginan kita, namun tidak ada satu strategi terbaik untuk *caching* konten dinamis, dan ada banyak situasi yang bisa mempengaruhi strategi *caching* [5].



Gambar 1. *Service worker* memiliki daur hidup (*life cycle*) yang berbeda dengan halaman web

D. Aplikasi Serupa

Aplikasi yang dibangun dalam studi ini tergolong jenis aplikasi web kursus *online* interaktif dengan materi *web development*. Ada banyak aplikasi serupa yang sudah ada antara lain Codecademy [6] dan FreeCodeCamp [7].

Kedua aplikasi ini sama-sama menyediakan berbagai materi *web development* dengan gratis, cukup dengan mendaftarkan akun pada web tersebut, hanya saja pada Codecademy tersedia fitur premium dimana pengguna bisa membayar untuk mendapatkan fitur lebih, sedangkan FreeCodeCamp tidak berbayar sama sekali dan merupakan aplikasi *open source*.

Dari sisi arsitektur, aplikasi pada studi ini berbeda dalam hal kerangka kerja *back-end* yang digunakan, kedua aplikasi tersebut menggunakan JavaScript. Pada sisi antarmuka memiliki kemiripan dimana ada deskripsi dan perintah pada halaman menjawab soal berada di sisi kiri, *code editor* pada bagian tengah, dan tampilan *browser* pada sebelah kanan.

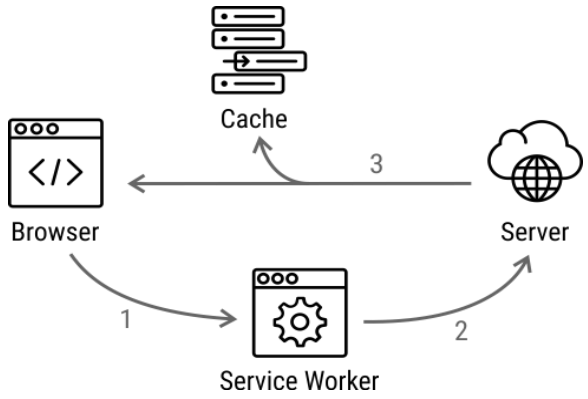
Dari sisi dukungan untuk mengerjakan tugas secara *offline*, pada FreeCodeCamp ketika sebuah soal sudah dibuka, penilaian bisa dijalankan secara *offline*, tetapi tidak bisa berpindah ke halaman lain, sedangkan pada Codecademy, pengguna diharuskan untuk selalu terkoneksi dengan internet, dan ketika *offline* penilaian tidak bisa dijalankan dengan indikasi bertuliskan *'Lost Connection to Codecademy'*.

III. ANALISIS DAN PERANCANGAN SISTEM

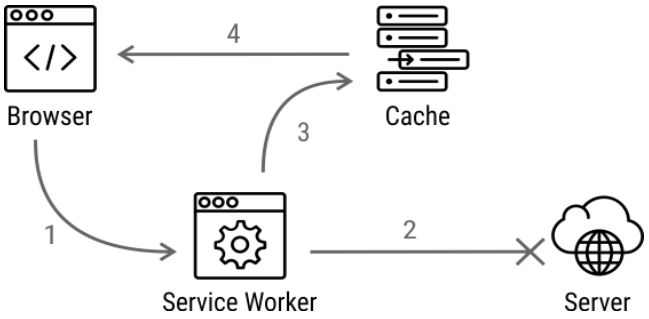
A. Perancangan Kemampuan Offline

Ada 3 komponen utama agar semua aplikasi web bisa dikatakan sebagai PWA yaitu: app shell, app manifest, dan service worker. Disini service worker memiliki peranan paling banyak di dalam konsep PWA. Service worker bekerja sebagai pengatur event fetch dari browser, lalu service worker memutuskan apakah request akan diteruskan ke server atau ke cache berdasarkan kondisi network online atau offline.

Dari banyak jenis strategi *caching* pada *service worker* akan digunakan jenis *network first, cache fallback.* Pada strategi ini pertama-tama *service worker* mengecek apakah *network* memberikan respons, dan jika berhasil, kembalikan data sekarang ke halaman. Jika gagal, *service worker* mengembalikan data dari *cache*. Strategi ini dipakai ketika membutuhkan data yang selalu baru seperti respons API, tapi butuh menampilkan sesuatu ketika *network* tidak bisa dicapai.



Gambar 2. *Service worker* dalam Kondisi *Offline*



Gambar 3. *Service Worker* dalam Kondisi *Online*

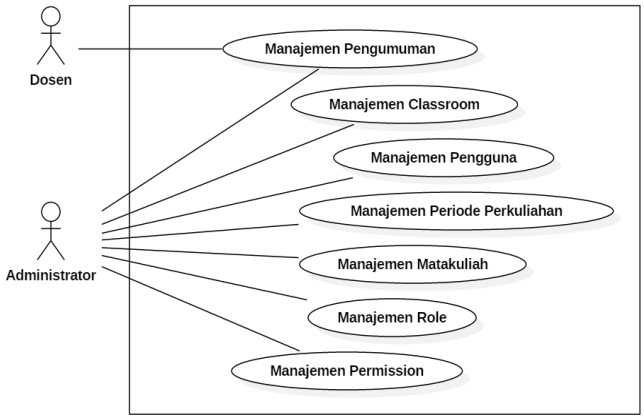
Service worker akan meneruskan setiap request berjenis GET dari halaman web ke server dalam kondisi online, lalu menduplikasi respons server dan disimpan ke dalam cache di browser, lalu respons server diteruskan kembali ke halaman web seperti digambarkan pada Gambar 2. Setiap halaman yang pernah dikunjungi oleh pengguna secara otomatis akan tersedia dalam versi offline-nya, hanya saja pengguna tidak bisa mengirim data ke server.

Ketika kondisi offline atau tidak ada respons dari server, maka akan dikembalikan halaman dari cache. Halaman bisa ditampilkan apabila halaman tersebut sudah ada di dalam cache sebelumnya seperti digambarkan pada Gambar 3.

B. Perancangan Kasus Penggunaan

Kasus penggunaan mewakili kebutuhan fungsional sistem. Perancangan diagram kasus penggunaan ini dibagi berdasarkan pengelompokan tugas-tugas dari pengguna yang ada. Pengguna dibagi berdasarkan role administrator, dosen, dan mahasiswa. Administator bertugas untuk melakukan manajemen-manajemen yang mendukung sistem dan proses pembelajaran diluar kelas. Sedangkan dosen dan mahasiswa, yang merupakan pusat dari segala tujuan diadakannya aplikasi ini, melakukan kegiatan-kegiatan yang berhubungan dengan proses pembelajaran. Oleh karena itu, berkaitan dengan proses pembelajaran yang ditujukan ke mahasiswa, kasus penggunaan dibagi menjadi kasus penggunaan non-pembelajaran dan kasus penggunaan pembelajaran.

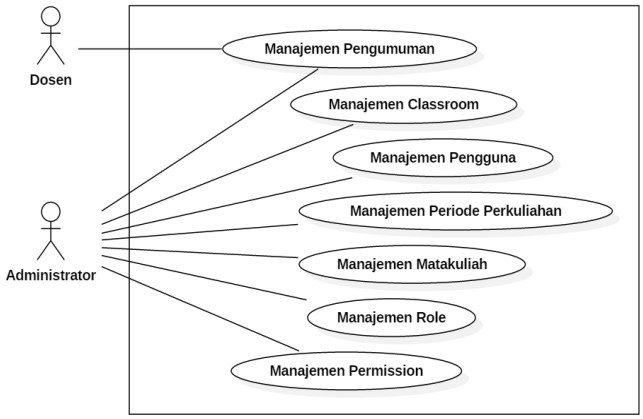
Diagram Kasus Penggunaan non-Pembelajaran ditunjukkan pada Gambar 4. Kasus penggunaan ini tidak melibatkan



Gambar 4. Kasus Penggunaan Non-Pembelajaran

mahasiswa dalam alur prosesnya, dan dititikberatkan pada aktor Administrator, sedangkan dosen hanya mendapat manajemen pengumuman. Semua Kasus Penggunaan ini tidak didukung sepenuhnya dengan kemampuan *offline*. Misalnya, aktor tetap bisa melihat daftar pengumuman dalam kondisi *offline*.

Diagram Kasus Penggunaan Pembelajaran ditunjukkan pada Gambar 5. Kasus penggunaan yang membutuhkan kemampuan *offline* yaitu kasus penggunaan dengan aktor mahasiswa yaitu kasus penggunaan melihat kelas yang diikuti, melihat daftar kuis, melihat daftar soal dari kuis, dan menjawab soal.



Gambar 4. Kasus Penggunaan Non-Pembelajaran

Kasus penggunaan dengan aktor dosen tidak didukung kemampuan *offline* secara khusus. Pengguna tetap bisa melihat kelas, kuis, atau soal sebagai fitur bawaan dari *service worker*, tetapi dosen tidak bisa menambah, mengubah, atau menghapus kelas, kuis, atau soal dalam kondisi *offline*.

IV. IMPLEMENTASI

A. Implementasi Kemampuan Offline

Implementasi kemampuan *offline* berpusat pada *service worker* dan terdiri dari berkas JavaScript *service worker* dan berkas JavaScript untuk meregistrasi *service worker*. *Service worker* digunakan untuk menangani koneksi, respons dan *request* maka harus diimplementasikan pada *domain* dengan protokol HTTPS agar terhindar dari pembajakan.

Pada awal pengembangan *e-Learning* ini, *service worker* yang diimplementasikan menggunakan strategi *caching cache first, network fallback*. Strategi ini dipilih karena waktu responsnya yang cepat. Setiap *Request* yang diminta oleh halaman, dilakukan pengecekan terlebih dahulu apakah ada *Request* yang sama di dalam *cache*, jika ada maka akan dikembalikan *Response* dari *cache*. Ternyata strategi *cache first* memiliki kekurangan yang fatal karena *token* Laravel yang digunakan oleh Laravel untuk mengenali *Request* ke server adalah dari *Request* sebelumnya, maka sering terjadi eror pada *form* yang ada di aplikasi, terutama *login*.

Pada versi *service worker* selanjutnya digunakan *service worker* seperti yang telah dirancang sebelumnya, strategi *service worker* yang digunakan adalah *network first, cache fallback*. *Service worker* yang baru ini sudah tidak mengalami kesalahan pada *token* Laravel, tetapi pengembalian respons ke halaman membutuhkan waktu yang lama karena menunggu *Response* dari server, atau menunggu sampai server tidak memberikan *Response*, barulah mengambil *Response* dari

cache. Untuk mengatasi hal ini ditambahkan satu buat pilihan *fetch* data yaitu ketika data yang dimintakan adalah data statis yang sudah ada di dalam *cache* yang bertipe JavaScript, CSS, gambar, atau *font* maka *Response* akan langsung dikembalikan dari *cache*.

V. PENGUJIAN DAN EVALUASI

A. *Pengujian Menggunakan Lighthouse*

Untuk menguji kebutuhan non-fungsional, pengujian dilakukan menggunakan *extension* Google Chrome yaitu Lighthouse [8]. Lighthouse adalah aplikasi *open source* yang dibangun oleh Google untuk menguji konsep PWA dengan aspek-aspeknya, *performance*, *accessibility* dan *best practices*. Lighthouse dapat dijalankan dari *browser* pengguna berupa ekstensi pada Google Chrome.

Pengujian menggunakan Lighthouse hanya bisa dijalankan untuk satu pengujian pada satu *browser client*. Untuk melakukan pengujian menggunakan Lighthouse pada banyak *client* digunakan layanan *third-party* Calibre [9]. Pengujian pada Calibre dilakukan secara otomatis menggunakan *emulation* yang menyerupai penggunaan *client* pada *browser* Chrome.

Hasil pengujian menggunakan Lighthouse, aplikasi web yang dibangun bisa dikatakan sudah memenuhi kriteria sebuah *Progressive Web Apps* dengan nilai 100 dari 100.

Aplikasi juga sudah mendapat nilai yang baik pada ketiga kriteria lainnya, kriteria *performance* dengan rata-rata 85 dari 100, kriteria *accessibility* dengan rata-rata 97 dari 100, dan kriteria *best practices* dengan rata-rata 100 dari 100.

Simulasi menggunakan bantuan Calibre memiliki sedikit kekurangan karena tidak menggambarkan berbagai *device* dan jaringan yang mungkin digunakan pengguna pada praktik sebenarnya karena pengujian dilakukan dalam *emulation*.

B. *Pengujian Menggunakan Chrome DevTools*

Untuk menguji kinerja *service worker* pada platform *e-Learning* digunakan Chrome DevTools dengan berbagai kondisi. Pengujian ini bertujuan untuk menunjukkan kinerja *service worker* pada aplikasi web dengan melihat jumlah *request*, waktu yang dibutuhkan untuk *load* semua konten halaman, dan jumlah data yang dikirim. Contoh hasil pengujian menggunakan *Chrome DevTools* bisa dilihat pada Gambar 6.

Pada skenario pertama halaman akan diuji dengan kondisi:

- Kunjungan Halaman : pertama kali
- Service Worker : belum ada
- Cache halaman : belum ada
- Internet : online

Hasil pengujian skenario pertama diketahui ada 64 *request* yang dibuat dari halaman, 1,1 MB data diterima, waktu memuat halaman 1,8 detik, dan selesai membuat semua *request* pada detik ke 3,18. Jumlah *request* yang banyak ini berasal dari *service worker* dimana semua berkas statis diunduh dan disimpan ke dalam *cache*. Proses ini terjadi dibelakang layar dan setelah halaman selesai dimuat sehingga tidak mengganggu pengguna dalam menggunakan aplikasi.

Pada skenario kedua halaman akan diuji dengan kondisi:

- Kunjungan Halaman : setelah pertama
- Service Worker : sudah aktif
- Cache halaman : sudah ada
- Internet : online

Hasil pengujian skenario kedua diketahui ada 10 *request* yang dibuat dari halaman, 0 B data diterima, waktu memuat halaman 1,32 detik, dan selesai membuat semua *request* pada detik ke 1,68. Jumlah *request* yang dibuat halaman yang sama setelah *service worker* aktif jauh lebih sedikit dari skenario pertama karena *service worker* tidak mengunduh data baru lagi untuk berkas statis. Pada skenario ini tidak ada data yang diterima oleh halaman karena semua data sudah ada di dalam *cache* dan dikembalikan ke halaman oleh *service worker* sehingga waktu memuat halaman lebih cepat 0,48 detik atau 26,6%.

Pada skenario ketiga halaman akan diuji dengan kondisi:

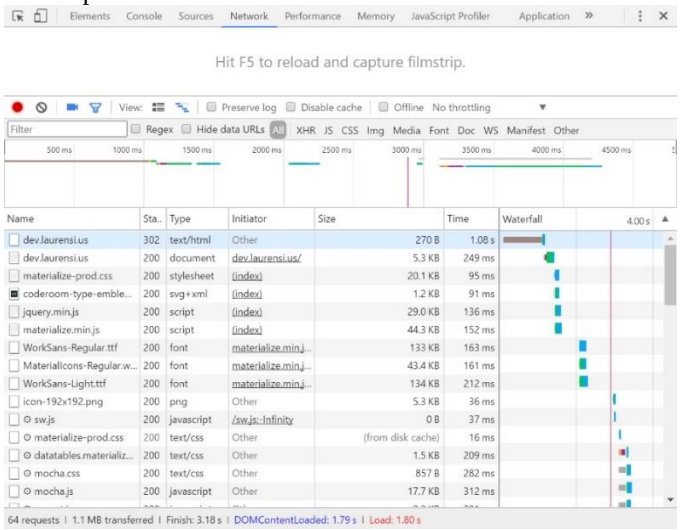
- Kunjungan Halaman : setelah pertama
- Service Worker : sudah aktif
- Cache halaman : belum ada
- Internet : online

Hasil pengujian skenario ketiga diketahui ada 11 *request* yang dibuat dari halaman, 970 B data diterima, waktu memuat halaman 1,86 detik, dan selesai membuat semua *request* pada detik ke 2,36. Jumlah *request* yang dibuat bertambah satu karena melalui proses login terlebih dahulu. Data yang diterima bertambah karena halaman yang dikunjungi tidak tersedia di dalam *cache* sehingga *request* diteruskan ke server dan data yang diterima hanyalah data HTML halaman baru tersebut, sedangkan berkas statis menggunakan data dari *cache*.

Pada skenario keempat halaman akan diuji dengan kondisi:

- Kunjungan Halaman : setelah pertama
- Service Worker : sudah aktif
- Cache halaman : belum ada
- Internet : online

Hasil pengujian skenario keempat diketahui ada 11 *request* yang dibuat dari halaman, 0 B data diterima, waktu memuat halaman 0,959 detik, dan selesai membuat semua *request* pada detik ke 5,29. Pengujian ini menunjukkan aplikasi dapat berjalan dalam kondisi offline. Pada pengujian ini aplikasi memuat konten ke halaman jauh lebih cepat dari sebelumnya karena tidak menunggu respons dari server sama sekali, dimana semua respons dikembalikan dari dalam *cache*.



Gambar 6. Hasil *Network Response* pada *Chrome DevTools*

VI. KESIMPULAN DAN SARAN

A. Kesimpulan

Selama proses perancangan, implementasi, dan pengujian dapat diambil kesimpulan sebagai berikut:

1. Platform *e-Learning* dengan penerapan konsep Progressive Web Apps mampu menampilkan halaman secara *offline* tetapi tidak bisa menyimpan, mengubah, atau menghapus data pada basis data.
2. Proses penilaian bisa dilakukan dalam kondisi *offline* dan memenuhi kebutuhan.
3. Platform *e-Learning* sudah memenuhi kebutuhan fungsional.
4. Hasil pengujian menggunakan Lighthouse menunjukkan rata-rata nilai 100 pada kriteria *Progressive Web Apps*, 85 pada kriteria *performance*, 97 pada kriteria *accessibility*, dan 100 pada kriteria *best practices*.
5. Waktu *loading* halaman lebih cepat 26.6% dengan *cache* dan *service worker*.

B. Saran

Saran untuk pengembangan dan perbaikan sistem di masa yang akan datang, di antaranya adalah sebagai berikut:

1. Aplikasi bisa dimodifikasi menggunakan arsitektur Client Side Rendering menggunakan kerangka kerja front-end JavaScript seperti Vue JS, Angular JS, atau React JS untuk aplikasi yang lebih cepat dan menghemat pengiriman data menggunakan JSON.
2. Aplikasi bisa dibuat mampu menangani request selain GET dalam kondisi offline dengan menunda request sampai sistem kembali online.
3. Modul penilaian menggunakan jquery expect masih harus menulis manual, bisa ditambahkan parser untuk suggestion ketika menulis kunci jawaban.

DAFTAR PUSTAKA

[1] "Your First Progressive Web App | Web," *Google Developers*. [Daring]. Tersedia pada: https://developers.google.com/web/fundamentals/getting-started/codelabs/your-first-pwapp/. [Diakses: 05-Jun-2017].

[2] "Can I use... Support tables for HTML5, CSS3, etc." [Daring]. Tersedia pada: https://caniuse.com/#feat=serviceworkers. [Diakses: 05-Jun-2017].

[3] "Cache," *Mozilla Developer Network*. [Daring]. Tersedia pada: https://developer.mozilla.org/en-US/docs/Web/API/Cache. [Diakses: 06-Jun-2017].

[4] D. Walsh, "Cache API," *David Walsh Blog*, 14-Feb-2016. .

[5] "Introduction to Progressive Web App Architectures | Caching Strategies Supported by sw-toolbox." [Daring]. Tersedia pada: https://developers.google.com/web/ilt/pwa/introduction-to-progressive-web-app-architectures#caching_strategies_supported_by_sw-toolbox. [Diakses: 05-Jun-2017].

[6] "Codecademy - learn to code, interactively, for free," *Codecademy*. [Daring]. Tersedia pada: https://www.codecademy.com/. [Diakses: 20-Jun-2017].

[7] "Learn to code and help nonprofits," *Free Code Camp*. [Daring]. Tersedia pada: http://www.freecodecamp.com. [Diakses: 20-Jun-2017].

[8] "Lighthouse | Web | Google Developers." [Daring]. Tersedia pada: https://developers.google.com/web/tools/lighthouse/. [Diakses: 15-Jun-2017].

[9] "Calibre - Web performance monitoring." [Daring]. Tersedia pada: https://calibreapp.com/. [Diakses: 21-Jun-2017].