DESAIN WEB

Sesi 6. Progressive Web Apps (PWA) dan Service Workers

Luthfil Khairi, S.Kom., M.Cs.

Departemen Informatika
Universitas Andalas

PENDAHULUAN

- Setelah mempelajari dasar-dasar desain dan layout web, kita akan menjelajahi konsep Progressive Web Apps (PWA) dan Service Workers, yang memungkinkan pengembangan aplikasi web dengan pengalaman pengguna yang lebih baik dan fungsionalitas offline.

PROGRESSIVE WEB APP (PWA)

- Progressive Web App (PWA) adalah sebuah teknologi web yang menggabungkan kemampuan terbaik dari aplikasi web dan aplikasi native.
- PWA memungkinkan pengguna mengakses website layaknya sebuah aplikasi mobile, namun tanpa perlu mengunduh dari app store.
- Dengan kata lain, PWA adalah website yang terlihat dan terasa seperti aplikasi native.

DEFINISI DAN KARAKTERISTIK PWA

- **Definisi:** PWA adalah sebuah aplikasi web yang dibangun dengan teknologi web modern (HTML, CSS, JavaScript) dan dioptimalkan untuk memberikan pengalaman pengguna yang mirip dengan aplikasi native.
- **Karakteristik Utama PWA:**
  - **Responsif:** PWA dapat menyesuaikan tampilan dan fungsionalitasnya dengan berbagai ukuran layar, baik itu desktop, tablet, maupun smartphone.
  - **Koneksi offline:** PWA dapat berfungsi meskipun perangkat tidak terhubung ke internet, berkat fitur caching.
  - **Pengalaman seperti aplikasi:** PWA dapat ditambahkan ke homescreen perangkat, memiliki ikon aplikasi, dan dapat bekerja secara fullscreen.
  - **Push notification:** PWA dapat mengirimkan notifikasi push ke perangkat pengguna, seperti aplikasi native.
  - **Secure:** PWA harus diakses melalui HTTPS untuk memastikan keamanan data pengguna

KEUNTUNGAN MENGGUNAKAN PWA

- **Pengalaman pengguna yang lebih baik:** PWA menawarkan pengalaman yang lebih cepat, lebih responsif, dan lebih interaktif dibandingkan dengan website tradisional.
- **Pemasangan yang mudah:** Pengguna dapat dengan mudah menambahkan PWA ke homescreen perangkat mereka tanpa perlu melalui app store.
- **Biaya pengembangan yang lebih rendah:** PWA dibangun menggunakan teknologi web yang sudah familiar bagi banyak pengembang, sehingga biaya pengembangannya cenderung lebih rendah dibandingkan dengan aplikasi native.
- **Perawatan yang lebih mudah:** Pembaruan PWA dapat dilakukan secara otomatis, sehingga pengguna selalu mendapatkan fitur terbaru.
- **Jangkauan yang lebih luas:** PWA dapat diakses melalui berbagai perangkat dan browser, tanpa perlu membuat versi aplikasi yang berbeda-beda.

PERBEDAAN ANTARA PWA DAN APLIKASI NATIVE

Fitur	PWA	Aplikasi Native
Pengembangan	Menggunakan teknologi web (HTML, CSS, JavaScript)	Menggunakan bahasa pemrograman spesifik untuk platform (Swift, Kotlin, Java)
Distribusi	Melalui web browser	Melalui app store
Pembaruan	Otomatis, melalui browser	Manual, melalui app store
Akses ke hardware	Terbatas (kamera, geolocation, dll.)	Lebih luas (sensor, hardware khusus)
Offline	Terbatas, bergantung pada fitur caching	Lebih baik, dapat bekerja sepenuhnya offline
Pengalaman pengguna	Mirip aplikasi native, namun mungkin ada sedikit perbedaan	Optimal untuk platform tertentu

KOMPONEN UTAMA PWA

- Progressive Web App (PWA) terdiri dari beberapa komponen kunci yang bekerja sama untuk memberikan pengalaman pengguna yang mirip dengan aplikasi native.
- Tiga komponen utama PWA:
  - Web App Manifest
  - Service Workers
  - HTTPS

WEB APP MANIFEST

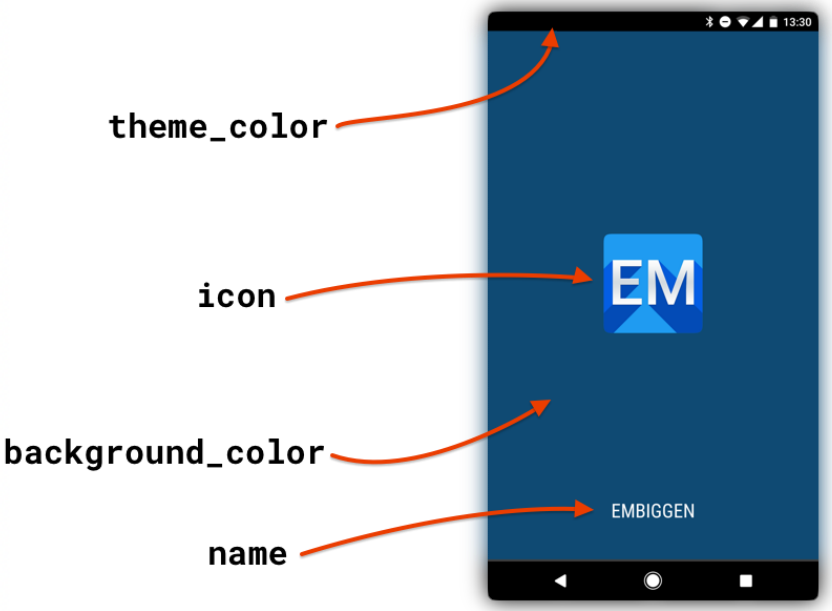
- Web App Manifest adalah file JSON yang berisi metadata tentang aplikasi web Anda.
- File ini memberikan informasi penting kepada browser tentang bagaimana aplikasi Anda harus ditampilkan dan berperilaku ketika ditambahkan ke homescreen perangkat pengguna

FUNGSI UTAMA WEB APP MANIFEST

- **Ikon aplikasi:** Menentukan ikon yang akan ditampilkan di homescreen.
- **Nama aplikasi:** Menentukan nama yang akan ditampilkan di homescreen.
- **Tema tampilan:** Menentukan tema warna dan tampilan antarmuka aplikasi.
- **Orientasi layar:** Menentukan orientasi layar yang didukung oleh aplikasi.
- **Layar awal:** Menentukan tampilan layar awal aplikasi.

CONTOH STRUKTUR WEB APP MANIFEST

```
{
  "name": "Your Great Site",
  "short_name": "Site 😎",
  "description": "Learn how to create and share something or other.",
  "start_url": "/?homescreen=1",
  "background_color": "#000000",
  "theme_color": "#0f4a73",
  "icons": [{
    "src": "icon-256.png",
    "sizes": "256x256",
    "type": "image/png"
  }]
}
```



PENJELASAN PROPERTI PENTING

- **name:** Nama lengkap aplikasi yang akan ditampilkan di homescreen.
- **short_name:** Nama pendek aplikasi yang mungkin digunakan dalam beberapa konteks.
- **icons:** Array yang berisi objek-objek ikon dengan berbagai ukuran.
- **start_url:** URL halaman awal aplikasi.
- **display:** Menentukan mode tampilan aplikasi (fullscreen, standalone, minimal-ui, browser).
- **theme_color:** Warna tema yang akan digunakan oleh browser.
- **background_color:** Warna latar belakang layar saat aplikasi sedang memuat.

Properti Tambahan (Beberapa di antaranya):

- **description:** Deskripsi singkat tentang aplikasi.
- **orientation:** Orientasi layar yang didukung (portrait, landscape).
- **scope:** Cakupan aplikasi (misalnya, direktori tertentu).

CARA MENGGUNAKAN WEB APP MANIFEST

- **Buat file:** Simpan file JSON ini dengan nama `manifest.json` di direktori root aplikasi Anda.
- **Tautkan dalam HTML:** Tambahkan tag link ke dalam `<head>` halaman HTML Anda:

```
HTML

<link rel="manifest" href="/manifest.json">
```

- **Uji:** Tambahkan aplikasi ke homescreen perangkat Anda dan lihat hasilnya.

SERVICE WORKERS

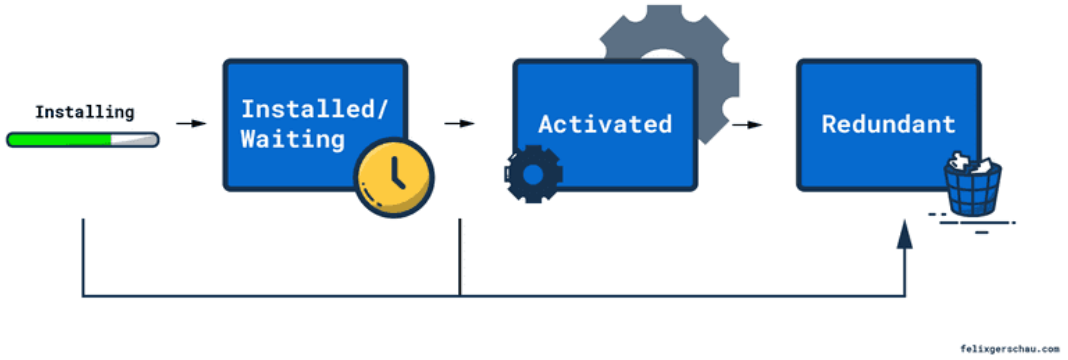
- Service Worker adalah script JavaScript yang berjalan di background, terpisah dari halaman web utama.
- Service Worker memungkinkan PWA untuk melakukan tugas-tugas yang tidak dapat dilakukan oleh JavaScript biasa, seperti:
  - **Caching:** Mengunduh dan menyimpan aset statis (HTML, CSS, JavaScript, gambar) sehingga aplikasi dapat berfungsi secara offline.
  - **Push notification:** Mengirimkan notifikasi push ke perangkat pengguna.
  - **Background sync:** Menjadwalkan tugas-tugas yang akan dijalankan ketika perangkat online kembali.
  - **Intersepsi permintaan:** Memanipulasi permintaan dan respons HTTP.

FUNGSI UTAMA SERVICE WORKER

- **Meningkatkan kinerja:** Dengan caching, Service Worker dapat mengurangi waktu loading halaman.
- **Menyediakan fitur offline:** Pengguna dapat mengakses sebagian atau seluruh fungsionalitas aplikasi meskipun tidak terhubung ke internet.
- **Memungkinkan push notification:** Mengirimkan notifikasi yang relevan kepada pengguna.

SIKLUS HIDUP SERVICE WORKER

- Siklus hidup Service Worker terdiri dari beberapa tahap:
  - **Registration:** Service Worker didaftarkan pada halaman web.
  - **Installing:** Service Worker diinstal dan cache awal dibuat.
  - **Activating:** Service Worker menjadi aktif dan menggantikan Service Worker yang lebih lama.
  - **Fetching:** Service Worker mencegat permintaan dan memberikan respons.



BROWSER SUPPORT DAN POLYFILLS

- **Browser Support:** Sebagian besar browser modern seperti Chrome, Firefox, Safari, dan Edge mendukung Service Worker. Namun, perlu diingat bahwa dukungan mungkin berbeda-beda antar versi browser.
- **Polyfills:** Polyfills adalah potongan kode JavaScript yang menambahkan fitur yang tidak didukung oleh browser tertentu. Meskipun ada beberapa polyfills untuk Service Worker, namun tidak disarankan untuk mengandalkan polyfills secara berlebihan, karena Service Worker adalah fitur yang cukup kompleks dan polyfills mungkin tidak mencakup semua kasus penggunaan.

CONTOH PENGGUNAAN SERVICE WORKER

JavaScript

```
// Daftar Service Worker
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('/service-worker.js')
      .then(registration
 => {
        console.log('Service Worker registered:', registration);
      })
      .catch(error => {
        console.error('Service Worker registration failed:',
 error);
      });
  });
}
```

Dalam kode tersebut:

- Kita memeriksa apakah browser mendukung Service Worker.
- Jika didukung, kita mendaftarkan Service Worker dengan file service-worker.js.
- Di dalam file service-worker.js, kita akan mengimplementasikan logika untuk meng-cache aset, menangani permintaan, dan fitur lainnya.

REGISTRASI SERVICE WORKER

- **Registrasi Service Worker** adalah proses menghubungkan sebuah halaman web dengan sebuah file JavaScript yang berisi logika Service Worker.
- Dengan kata lain, ini adalah langkah pertama untuk mengaktifkan fitur-fitur canggih yang ditawarkan oleh Service Worker, seperti caching, push notification, dan background sync

SYNTAX DASAR UNTUK REGISTRASI

JavaScript

```
if ('serviceWorker' in navigator) {
  window.addEventListener('load', () => {
    navigator.serviceWorker.register('/service-worker.js')
      .then(registration => {
        console.log('Service Worker
 registered:', registration);
      })
      .catch(error => {
        console.error('Service Worker registration failed:',
 error);
      });
  });
}
```

Penjelasan:

- **if ('serviceWorker' in navigator):** Memeriksa apakah browser mendukung Service Worker.
- **navigator.serviceWorker.register('/service-worker.js'):** Mendaftarkan Service Worker dengan file service-worker.js. Ganti /service-worker.js dengan path ke file Service Worker Anda.
- **.then():** Akan dijalankan jika registrasi berhasil.
- **.catch():** Akan dijalankan jika terjadi kesalahan selama registrasi.

MENANGANI KESUKSESAN DAN KEGAGALAN REGISTRASI

- **Kesuksesan:**
  - Anda bisa menampilkan pesan ke pengguna untuk menginformasikan bahwa Service Worker telah terdaftar.
  - Anda juga bisa melakukan tindakan lain, seperti mempersiapkan aplikasi untuk bekerja secara offline.
- **Kegagalan:**
  - Tampilkan pesan error kepada pengguna atau catat error ke dalam log untuk debugging.
  - Periksa apakah ada kesalahan dalam path file Service Worker, sintaks JavaScript, atau masalah jaringan.

UPDATING SERVICE WORKER

- Ketika Anda membuat perubahan pada file Service Worker, Anda perlu memperbarui Service Worker yang sudah terdaftar. Browser secara otomatis akan mengunduh versi baru dan menginstalnya, tetapi ada beberapa hal yang perlu diperhatikan:
  - **Siklus hidup Service Worker:** Saat Service Worker baru diinstal, ia tidak akan langsung aktif. Browser akan menunggu hingga semua tab yang menggunakan Service Worker yang lama ditutup, baru kemudian mengaktifkan Service Worker yang baru.
  - **Strategi caching:** Pastikan strategi caching Anda memungkinkan pembaruan konten secara efektif. Anda bisa menggunakan teknik seperti cache busting untuk memaksa browser mengunduh versi terbaru dari aset.
  - **Event `install` dan `activate`:** Gunakan event ini di dalam file Service Worker untuk mengontrol proses instalasi dan aktivasi.

FETCH API DAN CACHING

Fetch API

- Fetch API adalah sebuah interface dalam JavaScript modern yang digunakan untuk membuat permintaan HTTP ke server.
- Ini menggantikan metode-metode lama seperti `XMLHttpRequest`.
- Fetch API menawarkan cara yang lebih sederhana dan powerful untuk melakukan permintaan network, termasuk mendapatkan data dari API, mengunggah file, dan banyak lagi.

KEUNGGULAN FETCH API

- **Promise-based:** Membuat kode lebih mudah dibaca dan dikelola.
- **Mendukung semua jenis request:** GET, POST, PUT, DELETE, dan lainnya.
- **Header request yang fleksibel:** Mudah untuk mengatur berbagai macam header request.
- **JSON response:** Secara default, response dikembalikan dalam format JSON, sehingga memudahkan pengolahan data.

CACHING

- Caching adalah teknik menyimpan data secara sementara di perangkat pengguna untuk meningkatkan performa aplikasi.
- Data yang sudah di-cache dapat diakses lebih cepat daripada mengunduhnya kembali dari server.
- Dalam konteks web, caching biasanya digunakan untuk menyimpan aset statis seperti HTML, CSS, JavaScript, dan gambar.

INTERCEPT NETWORK REQUESTS DENGAN FETCH API

- Fetch API memungkinkan kita untuk mencegat setiap permintaan jaringan yang dilakukan oleh browser.
- Ini sangat berguna untuk mengimplementasikan strategi caching.

```
fetch('/data.json')
  .then(response => {
    if (!response.ok) {
      throw new Error('Network response was not ok');
    }
    return response.json();
  })
  .then(data => {
    // Proses data yang didapatkan
    console.log(data);
  })
  .catch(error => {
    console.error('There has been a problem with your fetch operation:', error);
  });
```

STRATEGI CACHING

- Terdapat beberapa strategi caching yang umum digunakan:
  - **Cache First:** Semua permintaan akan dilayani dari cache terlebih dahulu. Jika data tidak ditemukan di cache, baru kemudian akan diunduh dari server dan disimpan ke cache.
  - **Network First:** Permintaan akan dilayani dari jaringan terlebih dahulu. Jika berhasil, respons akan disimpan ke cache untuk digunakan di masa mendatang. Jika gagal, data akan diambil dari cache jika tersedia.
  - **Stale While Revalidate:** Data yang sudah kadaluarsa (stale) akan tetap digunakan, tetapi permintaan baru akan dikirim ke server untuk mendapatkan data yang segar.

HTTPS

- HTTPS (Hypertext Transfer Protocol Secure) adalah protokol komunikasi yang menggunakan enkripsi untuk mengamankan data yang ditransmisikan antara server dan klien.
- HTTPS sangat penting untuk PWA karena:
  - **Keamanan data:** Melindungi data sensitif pengguna dari serangan man-in-the-middle.
  - **Kepercayaan pengguna:** Menunjukkan bahwa aplikasi Anda aman dan dapat diandalkan.
  - **Fitur PWA:** Banyak fitur PWA, seperti Service Worker dan push notification, memerlukan koneksi HTTPS.

MENGAPA HTTPS PENTING UNTUK PWA

- **Service Worker:** Service Worker hanya dapat terdaftar pada halaman yang diakses melalui HTTPS.
- **Push notification:** Browser membutuhkan koneksi HTTPS untuk mengirimkan push notification.
- **Add to homescreen:** Pengguna lebih cenderung menambahkan PWA ke homescreen jika aplikasi tersebut menggunakan HTTPS.

INSTALLABLE WEB APPS: MENJADIKAN WEB APP SEPERTI APLIKASI NATIVE

- **Installable Web App** adalah sebuah fitur dari Progressive Web App (PWA) yang memungkinkan pengguna untuk menambahkan sebuah website langsung ke homescreen perangkat mereka, baik itu smartphone, tablet, atau bahkan desktop.
- Dengan begitu, pengguna dapat mengakses website tersebut dengan cara yang sama seperti mereka mengakses aplikasi native

KRITERIA UNTUK MEMBUAT PWA DAPAT DIINSTAL

- Agar sebuah website dapat diinstal sebagai PWA, secara umum website tersebut harus memenuhi kriteria berikut:
  - **HTTPS:** Website harus menggunakan protokol HTTPS untuk memastikan keamanan data pengguna.
  - **Web App Manifest:** Website harus memiliki file manifest.json yang berisi metadata tentang aplikasi, seperti nama, ikon, dan tema warna.
  - **Service Worker:** Website harus memiliki Service Worker yang aktif untuk mendukung fitur offline, push notification, dan caching.
  - **Responsif:** Website harus dapat menyesuaikan tampilan dan fungsionalitasnya dengan berbagai ukuran layar.

MENAMBAHKAN TOMBOL "ADD TO HOME SCREEN"

- Secara default, browser modern akan secara otomatis mendeteksi apakah sebuah website memenuhi kriteria untuk menjadi PWA dan akan menampilkan prompt "Add to Home Screen" kepada pengguna.
- Namun, Anda juga dapat memicu prompt ini secara manual menggunakan JavaScript.

CARA MEMICU PROMPT SECARA MANUAL

- **Deteksi kemampuan browser:** Periksa apakah browser mendukung fitur PWA.
- **Tampilkan tombol:** Jika browser mendukung, tambahkan tombol "Add to Home Screen" pada halaman Anda.
- **Gunakan Deferred Prompt:** Gunakan API Deferred Prompt untuk memicu prompt instalasi pada saat yang tepat.

CONTOH KODE JAVASCRIPT

JavaScript

```javascript
let deferredPrompt;

window.addEventListener('beforeinstallprompt', (event) => {
  // Menyimpan event untuk digunakan nanti
  deferredPrompt = event;
});

// Fungsi untuk menampilkan prompt
function showInstallPrompt() {
  // Menampilkan prompt
  deferredPrompt.prompt();
  // Reset deferredPrompt setelah digunakan
  deferredPrompt = null;
}
```

BEST PRACTICES UNTUK PWA YANG DAPAT DIINSTAL

- **Fokus pada pengalaman pengguna:** Pastikan PWA Anda memberikan pengalaman yang cepat, responsif, dan intuitif.
- **Optimalkan untuk offline:** Gunakan Service Worker untuk memungkinkan pengguna mengakses sebagian besar fitur aplikasi meskipun tidak ada koneksi internet.
- **Gunakan push notification:** Kirimkan notifikasi yang relevan kepada pengguna untuk meningkatkan engagement.
- **Perhatikan ukuran file:** Minimalkan ukuran file untuk mempercepat waktu loading.
- **Uji secara menyeluruh:** Uji PWA Anda pada berbagai perangkat dan browser untuk memastikan kinerja yang optimal.
- **Perbarui secara berkala:** Terus perbarui PWA Anda dengan fitur-fitur baru dan perbaikan bug.

MANFAAT MEMBUAT PWA YANG DAPAT DIINSTAL

- **Peningkatan engagement pengguna:** Pengguna lebih cenderung menggunakan aplikasi yang terpasang di homescreen.
- **Pengalaman pengguna yang lebih baik:** PWA memberikan pengalaman yang lebih mirip dengan aplikasi native.
- **Visibilitas yang lebih tinggi:** PWA dapat ditemukan di app store atau melalui pencarian.
- **Fleksibilitas:** PWA dapat diakses melalui berbagai perangkat dan platform.

FUNGSIONALITAS OFFLINE PADA PWA

- Salah satu keunggulan utama PWA adalah kemampuannya untuk berfungsi secara offline.
- Ini berarti pengguna dapat mengakses sebagian besar fitur aplikasi, bahkan ketika koneksi internet mereka terputus.
- Fitur ini sangat berguna, terutama di daerah dengan koneksi yang tidak stabil atau ketika pengguna berada di area tanpa jaringan.

STRATEGI CACHING UNTUK PWA

- **Caching** adalah teknik menyimpan data secara lokal di perangkat pengguna, sehingga data tersebut dapat diakses dengan cepat tanpa perlu mengunduh ulang dari server.
- Pada PWA, caching umumnya dilakukan menggunakan Service Worker.

STRATEGI CACHING YANG UMUM

- **Cache First:** Semua permintaan akan dilayani dari cache terlebih dahulu. Jika data tidak ditemukan di cache, baru kemudian akan diunduh dari server dan disimpan ke cache.
- **Network First:** Permintaan akan dilayani dari jaringan terlebih dahulu. Jika berhasil, respons akan disimpan ke cache untuk digunakan di masa mendatang. Jika gagal, data akan diambil dari cache jika tersedia.
- **Cache Only:** Semua permintaan hanya akan dilayani dari cache. Strategi ini cocok untuk data yang jarang berubah.

MENGELOLA ASET OFFLINE

- **Aset yang Dapat Dicache:**
  - **HTML:** Halaman utama dan halaman-halaman internal.
  - **CSS:** Gaya untuk halaman.
  - **JavaScript:** Script yang diperlukan untuk menjalankan aplikasi.
  - **Gambar:** Gambar yang digunakan dalam aplikasi.
  - **Font:** Font yang digunakan dalam aplikasi.

STRATEGI PENGELOLAAN ASET

- **Cache dengan bijak:** Jangan mencache semua aset secara berlebihan, karena dapat menghabiskan ruang penyimpanan perangkat.
- **Atur masa berlaku cache:** Tentukan berapa lama data harus disimpan dalam cache sebelum dihapus.
- **Bersihkan cache secara berkala:** Hapus cache yang sudah tidak digunakan untuk menghemat ruang penyimpanan.
- **Perbarui cache secara otomatis:** Perbarui cache ketika ada perubahan pada server.

MEMBERIKAN FEEDBACK KEPADA PENGGUNA SAAT OFFLINE

- **Tampilkan pesan:** Informasikan kepada pengguna bahwa mereka sedang offline dan beberapa fitur mungkin tidak tersedia.
- **Nonaktifkan fitur yang membutuhkan koneksi:** Sembunyikan atau nonaktifkan fitur yang membutuhkan koneksi internet.
- **Simpan tindakan pengguna:** Simpan tindakan pengguna yang dilakukan saat offline dan kirimkan ketika koneksi kembali

BEST PRACTICES UNTUK FUNGSIONALITAS OFFLINE

- **Prioritaskan aset penting:** Cache aset yang paling penting untuk pengalaman pengguna saat offline.
- **Gunakan teknik kompresi:** Kompres aset untuk mengurangi ukuran file dan mempercepat waktu loading.
- **Pertimbangkan ukuran perangkat:** Hindari mencache terlalu banyak data pada perangkat dengan ruang penyimpanan terbatas.
- **Uji secara menyeluruh:** Uji fungsionalitas offline pada berbagai perangkat dan jaringan untuk memastikan semuanya berjalan dengan baik.

KESIMPULAN

- **PWA** memungkinkan website untuk berperilaku seperti aplikasi mobile, memberikan pengalaman yang lebih cepat, lebih responsif, dan lebih andal. Fitur-fitur seperti instalasi pada homescreen, kemampuan bekerja offline, dan push notification membuat pengguna merasa lebih terhubung dengan aplikasi.
- **Service Workers** adalah jantung dari PWA. Mereka memungkinkan kita untuk mengontrol perilaku jaringan aplikasi, mengelola cache, dan memberikan pengalaman offline yang seamless. Dengan Service Workers, kita dapat membangun aplikasi web yang lebih cepat, lebih efisien, dan lebih dapat diandalkan.

TERIMAKASIH!

TUGAS!