Progressive Web Apps

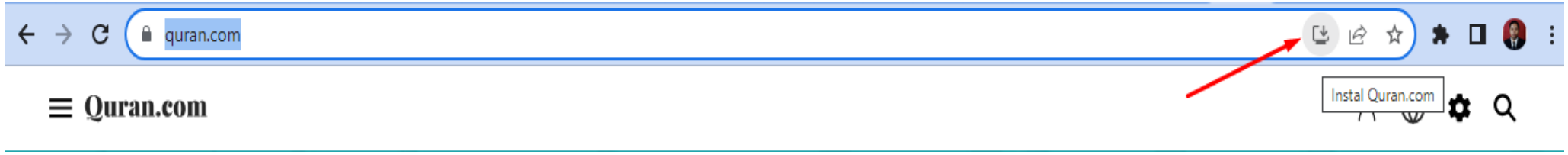
Oleh :

Harry Witriyono, M.Kom
NIDN. 0210126903

Fakultas Teknik, UM Bengkulu

Apa itu Progressive Web Apps ?

- Progressive Web Apps (PWA) adalah aplikasi web seperti biasanya tetapi dapat diinstal selayaknya aplikasi mobile pada perangkat mobile atau perangkat lainnya yang dibangun dengan bahasa pemrogramam web (HTML, CSS, JavaScript)
- PWA merupakan solusi cepat untuk menghasilkan aplikasi mobile yang dapat diakses lintas platform dan system operasi.
- Beberapa contoh aplikasi web dan perusahaan yang telah menggunakan PWA : https://quran.com, https://youtube.com , dan lain-lain
- Indikasi suatu aplikasi web berupa PWA, pada address bar di browser akan ada icon instalasinya.



Keunggulan PWA

1. **Responsif:** PWA dirancang untuk berfungsi dengan baik di berbagai perangkat, mulai dari desktop hingga ponsel pintar, menyesuaikan tampilan sesuai dengan ukuran layar yang berbeda.
2. **Offline Mode:** Salah satu fitur utama PWA adalah kemampuan untuk bekerja secara offline atau dengan koneksi internet yang terbatas. Mereka dapat menyimpan cache data untuk memungkinkan pengguna mengakses konten bahkan tanpa koneksi internet aktif.
3. **Fitur-Fitur Native-Like:** PWA bisa menggunakan fitur-fitur perangkat keras seperti kamera, sensor, dan bahkan notifikasi push, mirip dengan aplikasi native.
4. **Instalasi seperti Aplikasi:** Pengguna dapat "menginstal" PWA di layar beranda perangkat mereka, membuat ikonnya dan mengaksesnya dengan lebih cepat, tanpa harus melalui browser.
5. **Update Otomatis:** Seperti aplikasi web pada umumnya, PWA dapat diperbarui secara otomatis, tanpa memerlukan pengguna untuk mengunduh atau memperbarui aplikasi secara manual.

Kekurangan PWA

Beberapa kekurangan yang perlu dipertimbangkan:

1. **Keterbatasan Fungsionalitas:** PWA mungkin tidak memiliki akses penuh ke fitur-fitur perangkat keras pada tingkat yang sama seperti aplikasi native. Meskipun mereka dapat menggunakan beberapa fungsi seperti kamera atau notifikasi, ada beberapa fungsi perangkat keras yang mungkin tidak dapat diakses atau dioptimalkan sepenuhnya.
2. **Keterbatasan pada Platform Tertentu:** Meskipun banyak browser mendukung PWA, tidak semua platform atau browser mungkin memiliki dukungan yang sama. Ini dapat menyebabkan perbedaan dalam kinerja atau fungsionalitas PWA tergantung pada perangkat atau browser yang digunakan pengguna.
3. **Keterbatasan Konektivitas:** Meskipun PWA dapat berfungsi dalam mode offline, beberapa fitur atau konten mungkin memerlukan koneksi internet untuk diakses. Konten dinamis atau data baru mungkin tidak tersedia saat offline.
4. **Pemrosesan yang Terbatas:** Performa dan kinerja PWA dapat dipengaruhi oleh kemampuan pemrosesan perangkat pengguna. PWA yang memerlukan pengolahan berat mungkin tidak berjalan seefisien aplikasi native di perangkat dengan spesifikasi rendah.
5. **Kurangnya Kepahaman Pengguna:** Konsep PWA masih relatif baru, sehingga pengguna mungkin tidak sepenuhnya mengerti atau terbiasa dengan cara mengakses, menginstal, atau mengelola PWA di perangkat mereka.

kekurangan ini terus diperbaiki seiring dengan kemajuan teknologi web dan perangkat lunak, membuat PWA semakin berkembang dan meningkatkan kualitasnya seiring waktu.

Struktur Dasar PWA

1. **Service Worker:** Ini adalah bagian inti dari PWA. Service worker adalah skrip JavaScript yang berjalan di background dan memungkinkan PWA untuk melakukan banyak hal, seperti caching konten, menyediakan fungsi offline, dan mengelola notifikasi push. Service worker memungkinkan aplikasi untuk berfungsi bahkan tanpa koneksi internet aktif.
2. **Manifest File:** File manifest adalah file JSON yang berisi informasi tentang aplikasi, seperti nama, deskripsi, ikon, dan konfigurasi tampilan layar. Ini memungkinkan pengguna untuk "menginstal" PWA di perangkat mereka dan membuat ikonnya muncul di layar beranda, memberikan pengalaman yang lebih mirip dengan aplikasi native.
3. **HTTPS:** PWA harus di-host melalui koneksi HTTPS untuk memberikan keamanan yang diperlukan. Ini juga diperlukan untuk beberapa fitur PWA, seperti service worker, agar dapat berfungsi.
4. **Responsive Design:** Desain responsif memastikan bahwa PWA dapat menyesuaikan tampilannya dengan berbagai perangkat, mulai dari desktop hingga ponsel pintar, sehingga memberikan pengalaman pengguna yang konsisten.
5. **App Shell:** Ini adalah kerangka atau struktur dasar dari PWA yang memungkinkan konten utama untuk dimuat secara cepat saat pengguna mengakses aplikasi. App shell terdiri dari elemen-elemen antarmuka pengguna yang mendasar seperti header, footer, dan navigasi, yang di-cache untuk memberikan pengalaman yang responsif dan cepat.
6. **Notifikasi Push:** PWA dapat menggunakan fitur notifikasi push untuk berkomunikasi dengan pengguna, memberikan pemberitahuan tentang pembaruan atau informasi penting lainnya, meskipun pengguna tidak sedang membuka aplikasi.

Struktur Folder Aplikasi

Tidak ada standar baku, tetapi berikut ini adalah struktur folder yang umum digunakan:

Root Folder: berisi file index.html sebagai landing page aplikasi, manifest.json, service-worker.js, dan file favicon.ico

Sub Folder CSS yang berisi file-file CSS

Sub Folder JS yang berisi file-file Java Script

Sub Folder Images yang berisi file-file gambar

Sub Folder Pages yang berisi file-file halaman-halaman aplikasinya

Sub Folder Lib yang berisi file-file library

Sub Folder Asset yang berisi file-file lain.

Elemen File Manifest.json

- File manifest.json berisi informasi metadata tentang aplikasi PWA-nya.
- Elemen-elemennya :

```
{
 "name": "My Progressive Web App",
 "short_name": "MyPWA",
 "description": "Aplikasi yang luar biasa!",
 "icons": [
  {
   "src": "icon-48x48.png",
   "type": "image/png",
   "sizes": "48x48"
  },
  {
   "src": "icon-96x96.png",
   "type": "image/png",
   "sizes": "96x96"
  }
 ],
 "start_url": "/index.html",
 "display": "standalone",
 "background_color": "#ffffff",
 "theme_color": "#3367D6",
 "orientation": "any"
}
```

Elemen File service-worker.js

- File service-worker.js adalah file bagian inti dari Progressive Web Apps (PWA) yang mengatur perilaku cache, notifikasi, dan beberapa fungsi lainnya di latar belakang.
- Fungsi umum yang mungkin ada di dalamnya:

1. **Cache Management:** Service worker dapat digunakan untuk menyimpan cache dari aset-aset aplikasi seperti file HTML, CSS, JavaScript, gambar, dan data lainnya. Dengan cache, aplikasi dapat bekerja secara offline atau memuat lebih cepat saat pengguna kembali mengunjungi halaman yang sama.
2. **Strategi Cache:** Banyak service worker menggunakan berbagai strategi cache seperti cache-first, network-first, atau stale-while-revalidate untuk mengelola permintaan data dari cache atau server.
3. **Notifikasi Push:** Service worker dapat digunakan untuk mengatur notifikasi push kepada pengguna, memberi tahu mereka tentang pembaruan atau informasi penting bahkan saat aplikasi tidak terbuka.
4. **Event Listeners:** Service worker dapat mendengarkan event-event seperti ‘fetch’ untuk mengendalikan permintaan jaringan, ‘install’ untuk menginisialisasi cache, ‘activate’ untuk membersihkan cache lama, dan lainnya.

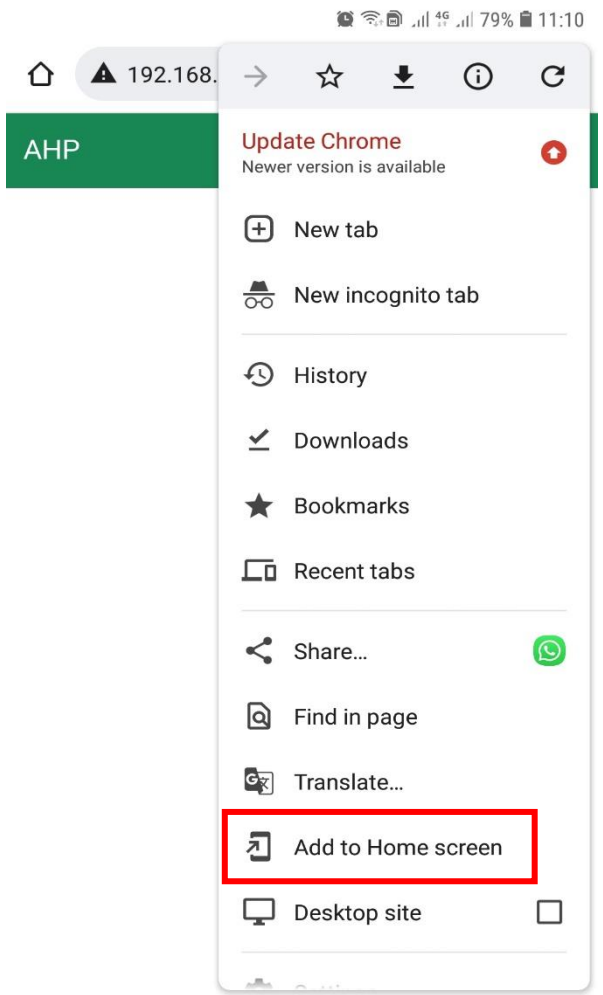
Isi file service-worker.js

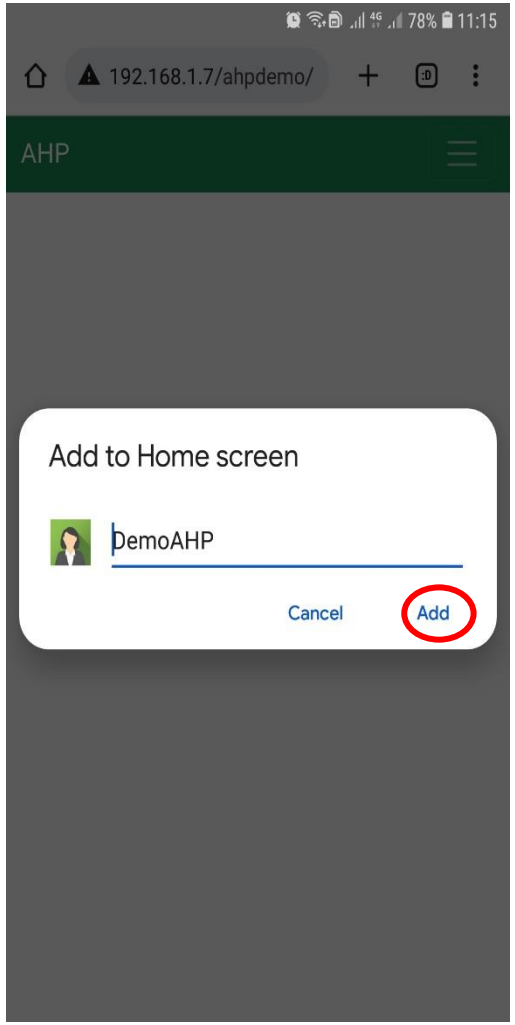
```
1  // Install service worker
2  self.addEventListener('install', event => {
3    event.waitUntil(
4      caches.open('app-cache').then(cache => {
5        return cache.addAll([
6          '/',
7          '/index.html',
8          '/styles/main.css',
9          '/scripts/main.js',
10         '/images/logo.png'
11         // ... daftar aset yang akan di-cache
12       ]);
13     })
14   );
15 });
```

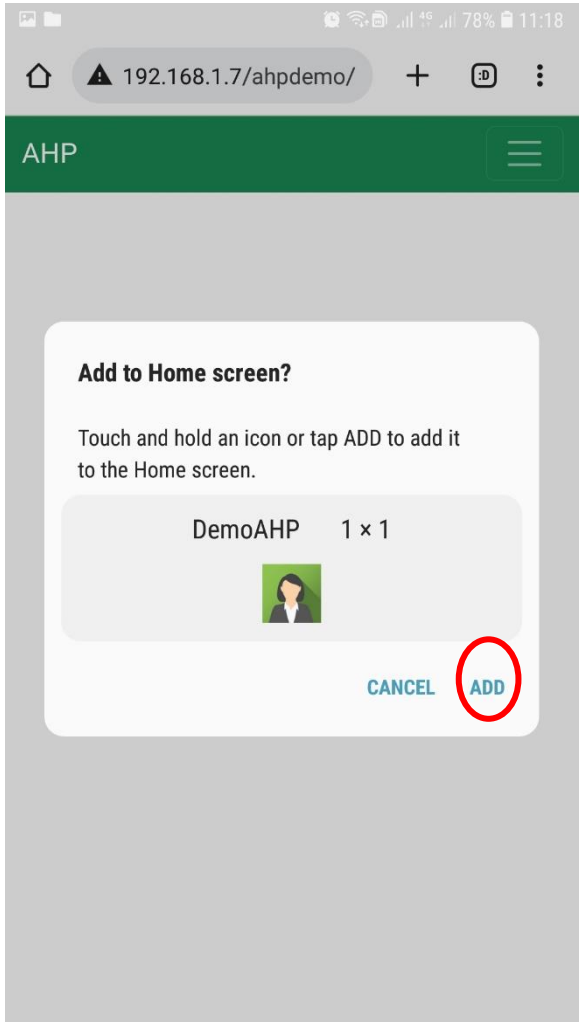
```
16
17  // Intercept network requests
18  self.addEventListener('fetch', event => {
19    event.respondWith(
20      caches.match(event.request).then(response => {
21        return response || fetch(event.request);
22      })
23    );
24  });
25
26  // Handle push notifications
27  self.addEventListener('push', event => {
28    const options = {
29      body: event.data.text(),
30      icon: '/images/icon.png'
31    };
```

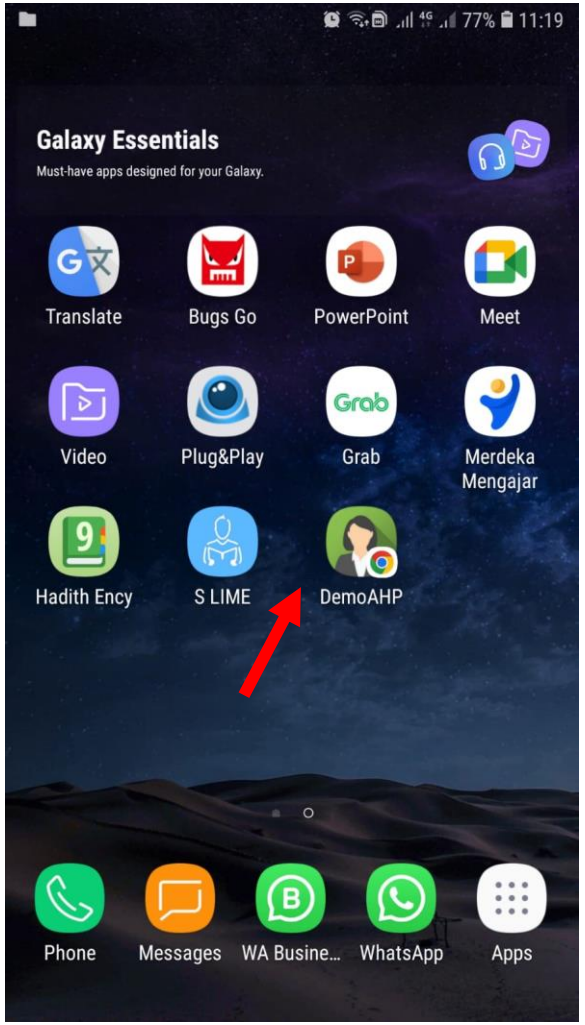
```
32
33    event.waitUntil(
34      self.registration.showNotification('PWA Notification', options)
35    );
36  });
37
```

Cara Install Aplikasi Model PWA di HP









Sesi Praktikum

1. Buat Struktur Folder :

1. Buat folder pdfpwa di htdocs
2. Buat sub folder assets di pdfpwa
3. Buat sub folder css di assets
4. Buat sub folder js di assets
5. Buat sub folder img di assets
6. Buat sub folder others di assets

2. Buat file sw.js di folder assets/js

- File service worker ini menangani pekerjaan pertama kali ke aplikasi PWA kita dan sebagai proxy antara aplikasi kita dengan jaringan baik saat sedang tersedia jaringan atau tidak.
- File ini menentukan apa yang akan dijadikan cache dari sumber aplikasi kita
- File ini diregistrasi pada index.php / index.html aplikasi di dalam bagian tag <body> dan </body> yang bentuknya seperti berikut ini:

```
<script> if (typeof navigator.serviceWorker !== 'undefined') {
navigator.serviceWorker.register('sw.js') } </script>
```

- Informasi lengkapnya dapat diakses di : https://docs.pwabuilder.com/#/home/sw-intro?id=overview

Isi file sw.js

```
const CACHE_NAME = 'cool-cache';

// Add whichever assets you want to precache here:
const PRECACHE_ASSETS = [
    '/assets/',
    'css/bootstrap.min.css',
    'js/bootstrap.bundle.min.js',
    'js/register.js',
    'others/BuatAplikasiCekStokGudang',
    'img/logo.png',
    '/src/'
]

// Listener for the install event - precaches our assets list on service worker install.
self.addEventListener('install', function(event) {
    event.waitUntil(
        caches.open(CACHE_NAME)
        .then(function(cache) {
            return cache.addAll(toCache)
        })
        .then(self.skipWaiting())
    )
})

self.addEventListener('fetch', function(event) {
    event.respondWith(
        fetch(event.request)
        .catch(() => {
            return caches.open(CACHE_NAME)
            .then((cache) => {
                return cache.match(event.request)
            })
```

```
        })
    )
})

self.addEventListener('activate', function(event) {
    event.waitUntil(
        caches.keys()
        .then((keyList) => {
            return Promise.all(keyList.map((key) => {
                if (key !== CACHE_NAME) {
                    console.log('[ServiceWorker] Hapus cache lama', key)
                    return caches.delete(key)
                }
            }))
        })
        .then(() => self.clients.claim())
    )
})
```

Penjelasan Kode

- Pada baris 6-10 adalah daftar file yang akan di-cache ke media tujuan jika diinstall nanti.

3. Buat file manifest.json

- File ini berfungsi memberikan informasi tentang aplikasi web yang akan diinstal nantinya.
- Posisi file di subfolder assets/js
- Ada banyak member untuk isi filenya paling tidak seperti contoh kode berikut ini.
- Informasi lengkapnya bisa diakses di : https://developer.mozilla.org/en-US/docs/Web/Manifest

Isi file manifest.json

```
{
"orientation": "any",
"name": "MyPDFBookV.2023",
"short_name": "MyPDFBook",
"start_url": "index.php",
"display": "standalone",
"theme_color": "#0d0072",
"background_color": "#0d0072",
"description": "A simple pdf book to make a web app.",
"icons": [
    {
      "src": "img/logo.png",
      "sizes": "192x192",
      "type": "image/png"
    }
  ]
}
```

4. Sertakan link manifest.json di file index.php

- Penyertaan link ini penting sekali sehingga aplikasi web kita akan menjadi Progressive Web Apps.
- Peletakanya diantara tag <head> dan </head> dengan bentuk tag seperti berikut ini:

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

5. Membuat file index.php

- File ini ditempatkan pada aplikasi folder kita pdfpwa sebagai file pertama yang diakses browser.
- Pada file ini harus ada link ke file manifest.json dan registrasi file sw.js

Isi File index.php untuk PWA

index.php

```php
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Buat Aplikasi Cek Stok Gudang</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="manifest" href="assets/manifest.json" />
  <link href="assets/css/bootstrap.min.css" rel="stylesheet">
  <script src="assets/js/bootstrap.bundle.min.js"></script>
</head>
<body>

<div class="container">
  <h1>Contoh PWA Single Page</h1>
  <p>
   <object
   type="application/pdf"
   data="assets/others/BuatAplikasiCekStokGudang.pdf"
   width="100%"
   height="600px">
   </object>
  </p>
</div>
<script>
  if (typeof navigator.serviceWorker !== 'undefined') {
    navigator.serviceWorker.register('sw.js')
  }
</script>
</body>
</html>
```

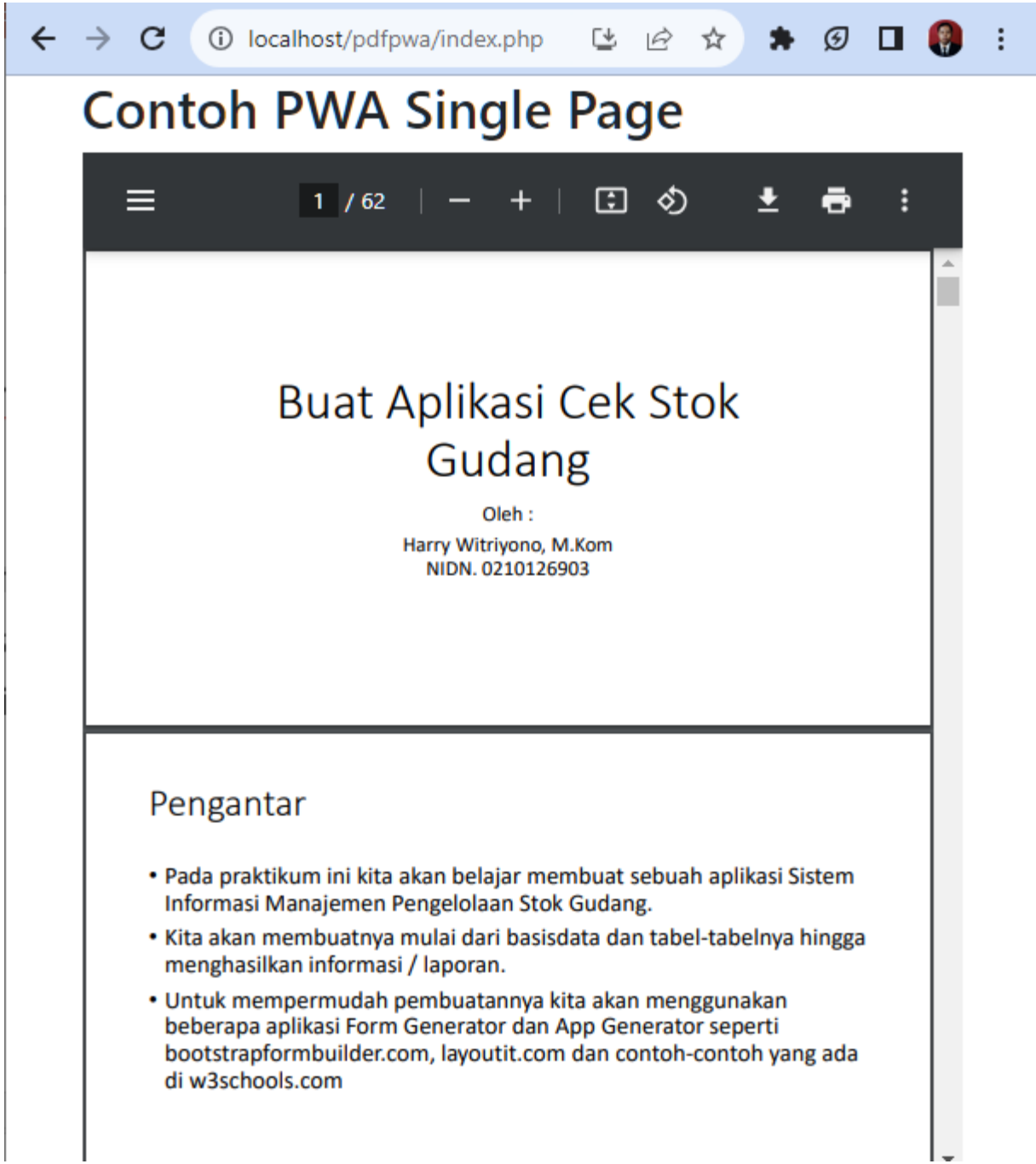
6. Copy file-file yang diperlukan ke foldernya

- Masukkan file bootstrap.min.css di sub folder assets/css
- Masukkan file bootstrap.bundle.min.js di sub folder assets/js
- Masukkan file logo.png ukuran 192x192 pixels di folder assets/img
- Masukkan file pdf pada praktikum ini BuatAplikasiCekStokGudang.pdf di sub folder assets/others

7. Jalankan Aplikasinya

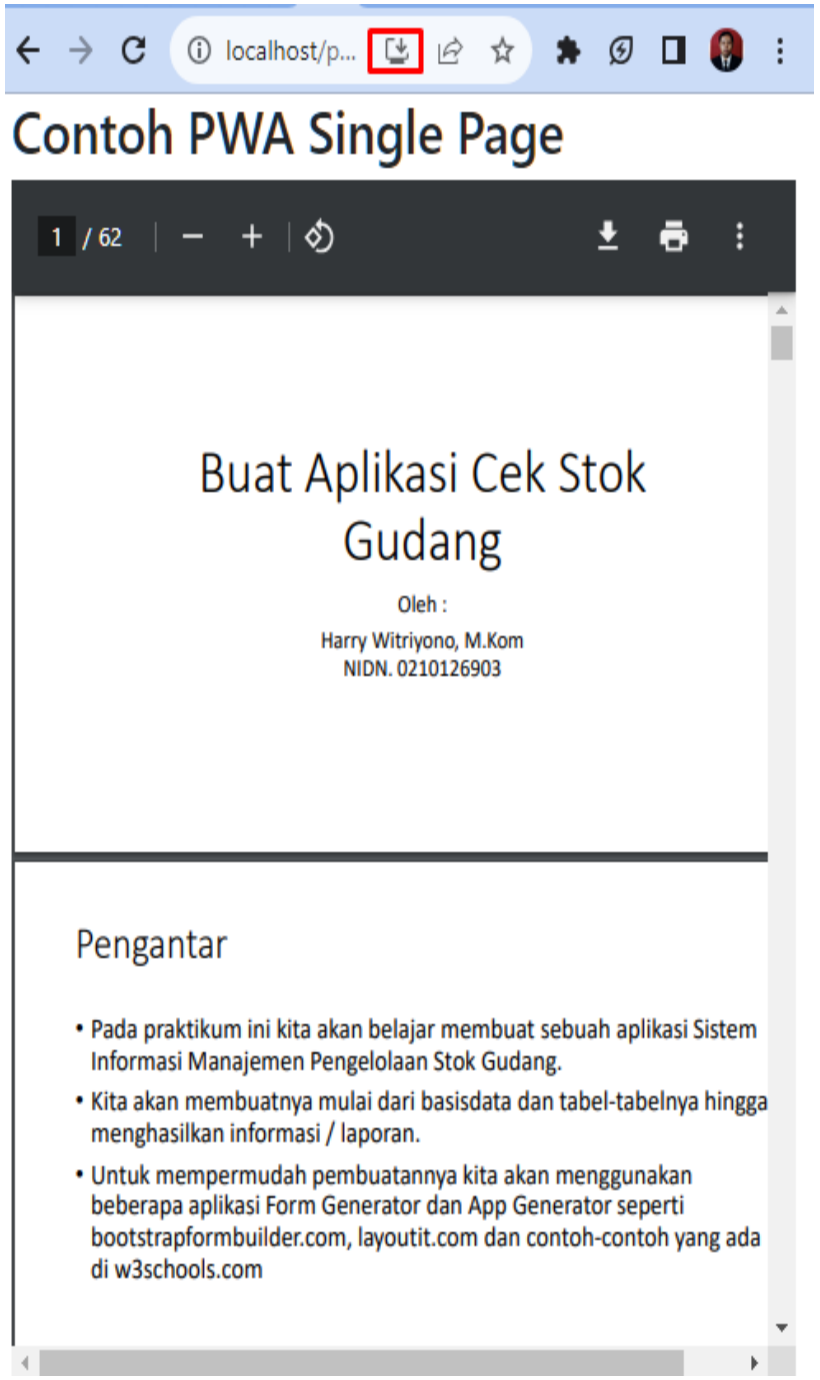
- Buka browser pada handphone atau pc anda dan akses :

  http://localhost/pdfpwa



8. Install aplikasi pdwpwa

- Klik icon install app di address bar browser anda di smartphone atau pc anda.



Sesi Praktikum 2. PWA Biodata

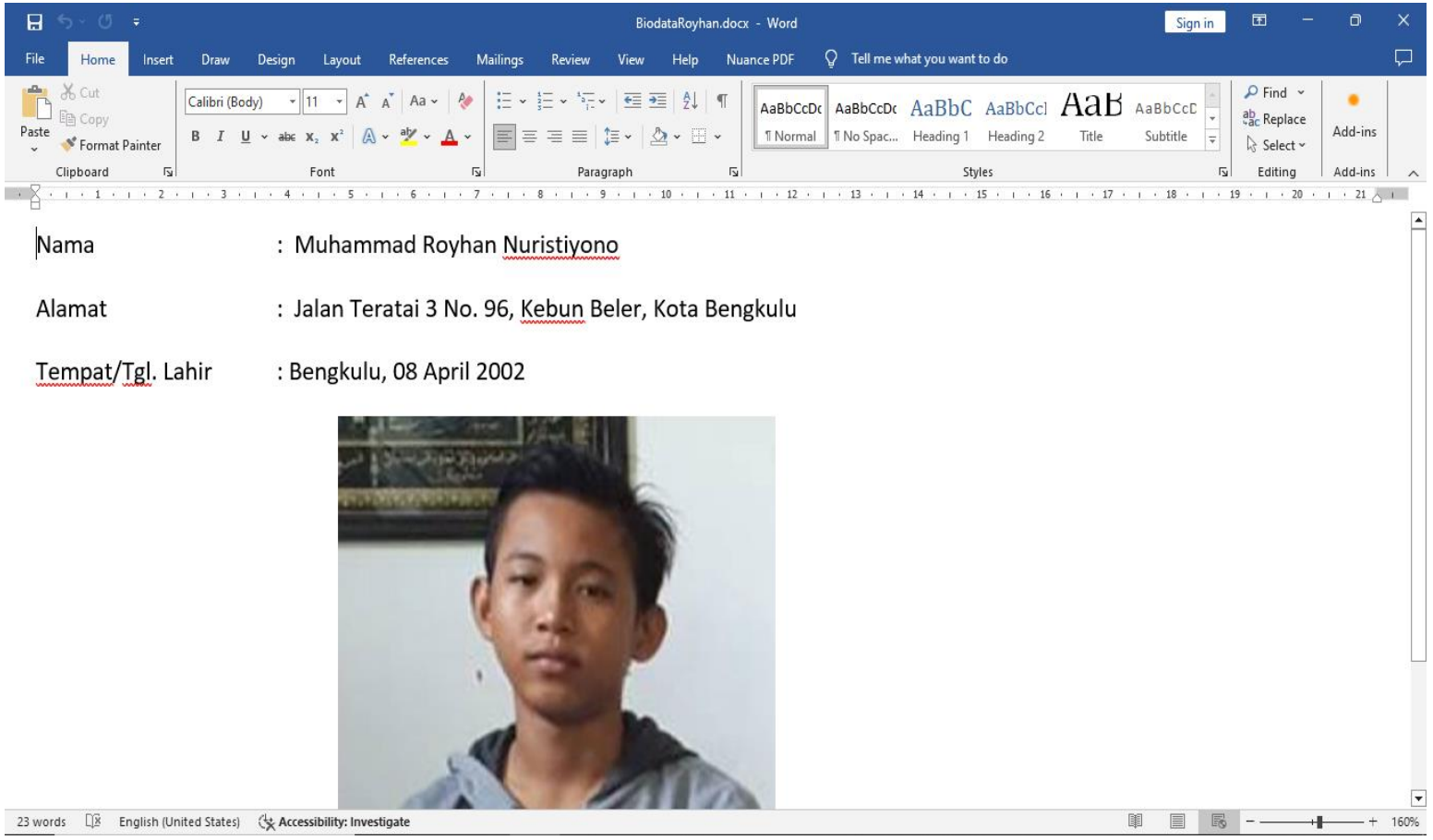
Untuk mudahnya silahkan gunakan fasilitas dari Microsoft Word untuk membuat halaman web index.html

1. Buat Folder Aplikasi di htdocs

- Buat folder aplikasi yang akan dibuat di folder public server apache anda di htdocs dengan nama : pwa
- Gunakan folder ini untuk menyimpan file index.html dan folder index_files yang dihasilkan dari pembuatannya nanti dengan Microsoft Word.

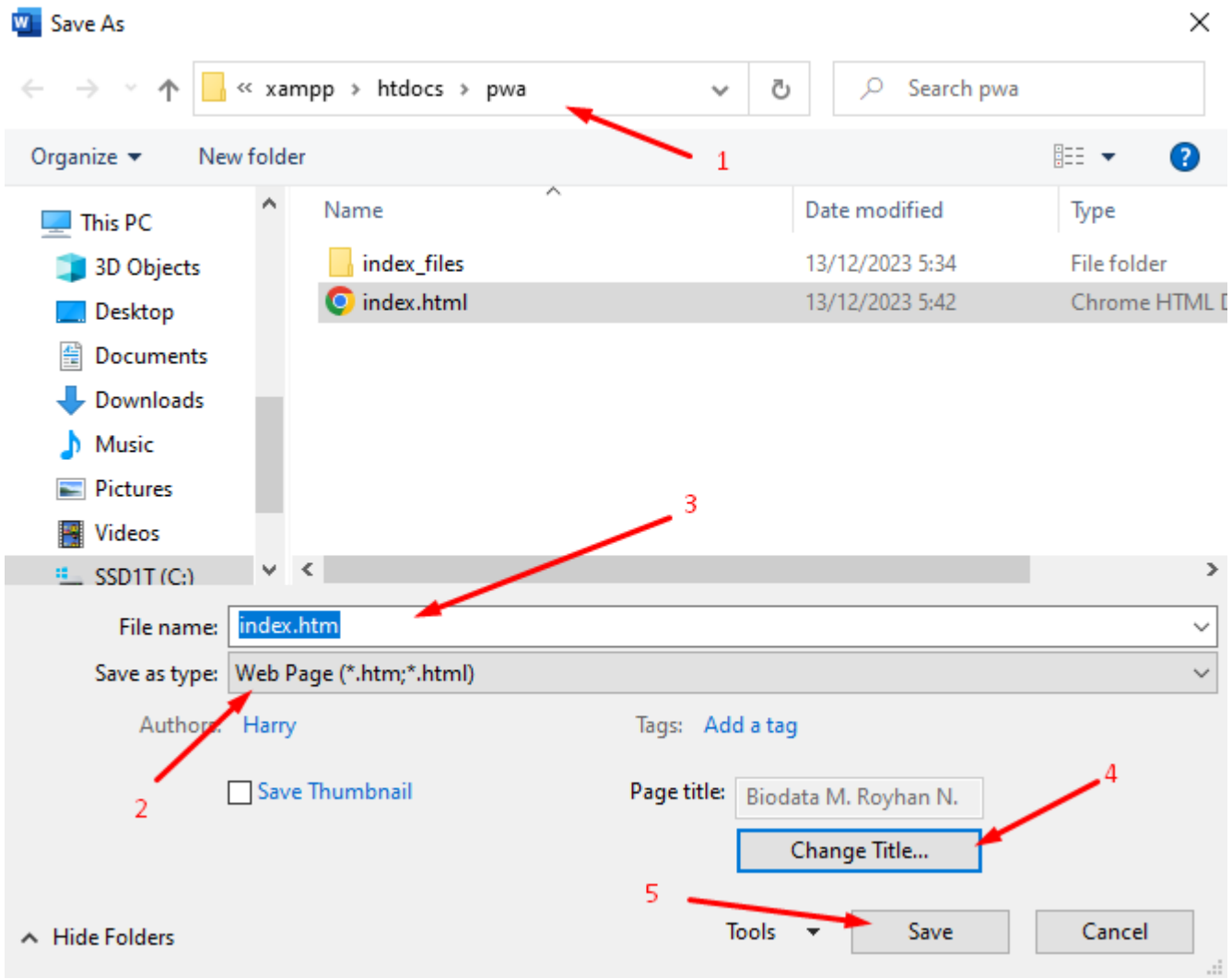
2. Buat File index.html dengan Microsoft Word

- Buat dokumen Microsoft Word seperti berikut ini :



3. Simpan menjadi web

- Ikuti tahapan seperti pada gambar untuk melaksanakannya.



4. Edit file index.html

- Edit file index.html yang dihasilkan dengan menambahkan tag seperti pada gambar jenis document dan link file manifest.json-nya

```
1  <!DOCTYPE html>
2  <html xmlns:v="urn:schemas-microsoft-com:vml"
3  xmlns:o="urn:schemas-microsoft-com:office:office"
4  xmlns:w="urn:schemas-microsoft-com:office:word"
5  xmlns:m="http://schemas.microsoft.com/office/2004/12/omml"
6  xmlns="http://www.w3.org/TR/REC-html40">
7
8  <head>
9  <meta http-equiv=Content-Type content="text/html; charset=windows-1252">
10 <meta name=ProgId content=Word.Document>
11 <meta name=Generator content="Microsoft Word 15">
12 <meta name=Originator content="Microsoft Word 15">
13 <link rel="manifest" href="index_files/manifest.json" />
14 <link rel=File-List href="index_files/filelist.xml">
15 <link rel=Edit-Time-Data href="index_files/editdata.mso">
```

1

2

5. Tambahkan Script Pengaktifan File Service Worker

- Letakkan pada bagian bawah dari tag <body> pada file index.html tadi tag <script> dan isinya hingga tag penutupnya </script> untuk file service worker sw.js yang memberikan service pada proses instalasi Progressive Web Apps.

```
<body lang=EN-ID style='tab-interval:36.0pt;word-wrap:break-word'>
<script> if (typeof navigator.serviceWorker !== 'undefined') {
navigator.serviceWorker.register('index_files/sw.js') } </script>

<div class=WordSection1>

<p class=MsoNormal><span lang=EN-US>Nama <span style='mso-tab-count:3'>
```

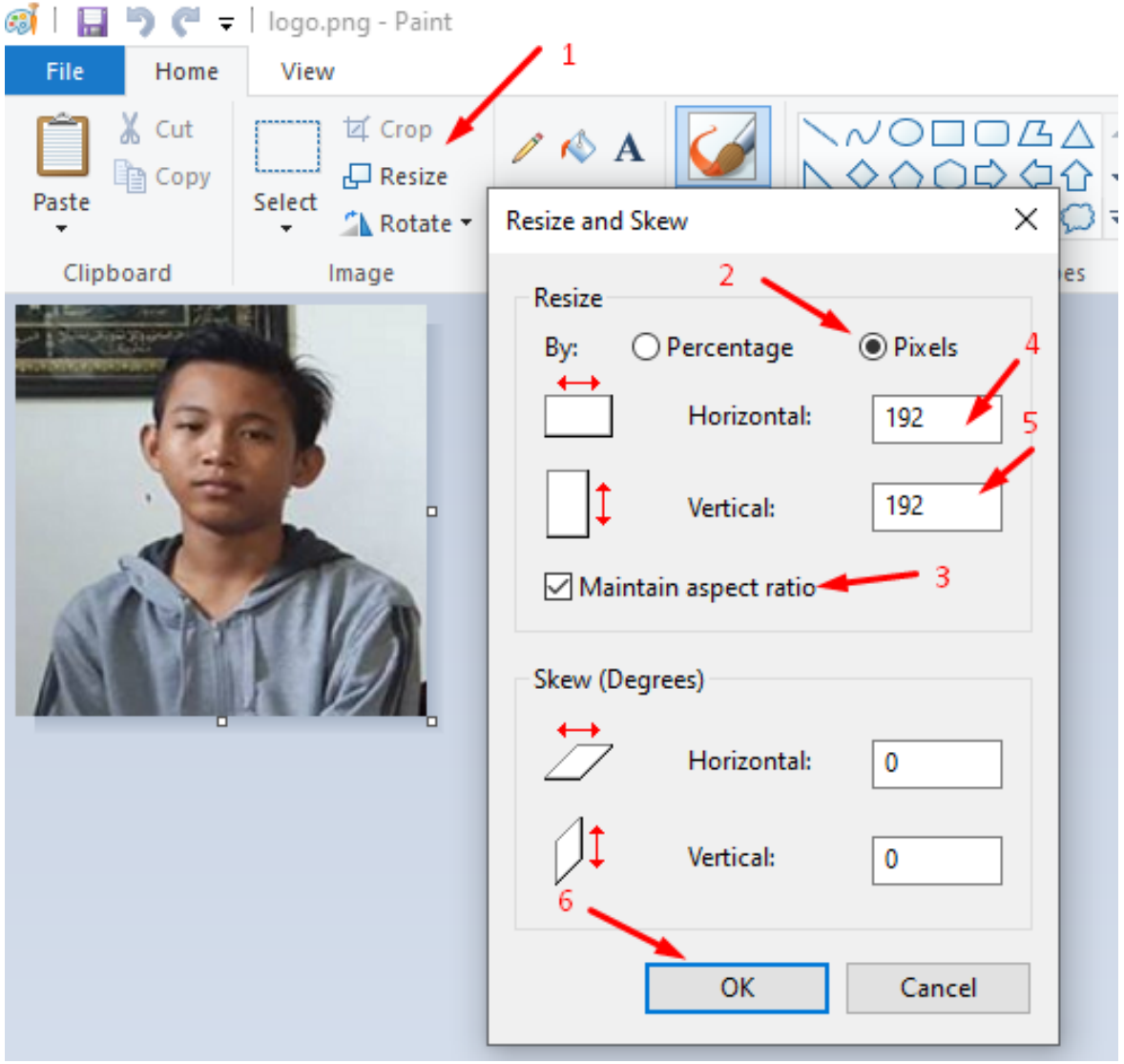
6. Buat File manifest.json

- Buat file manifest.json di folder index_files seperti pada gambar untuk identitas dari aplikasi PWA kita

```
{
"orientation": "any",
"name": "BiodataRoyhan",
"short_name": "BioRoy",
"start_url": "index.html",
"display": "standalone",
"theme_color": "#0d0072",
"background_color": "#0d0072",
"description": "Contoh sederhana membuat PWA.",
"icons": [
        {
          "src": "logo.png",
          "sizes": "192x192",
          "type": "image/png"
        }
    ]
}
```

7. Buat File icon.png

- Buat file icon.png berukuran 192x192 pixels dengan aplikasi PaintBrush atau sejenisnya.



8. Buat File register.js

- Buat file register.js dan simpan di dalam folder index_files untuk meregister file-file yang dibutuhkan pada aplikasi PWA kita

```
1   document.addEventListener('DOMContentLoaded', init, false);
2
3   function init() {
4       if ('serviceWorker' in navigator && navigator.onLine) {
5           navigator.serviceWorker.register('assets/js/sw.js')
6           .then((reg) => {
7               console.log('Registrasi service worker Berhasil', reg);
8           }, (err) => {
9               console.error('Registrasi service worker Gagal', err);
10          });
11      }
12  }
```

9. Buat File sw.js

- Buat file service worker sw.js dan simpan di folder index_files yang tadi telah kita masukkan script pemanggilnya di file index.html.
- File ini berfungsi memproses semua aktifitas aplikasi PWA kita.
- Yang perlu anda perhatikan adalah penggantian nama pada konstanta array CACHE_NAME dan penentuan lokasi dan file-file yang akan diinstalkan pada cache dari aplikasi PWA kita di konstanta array PRECACHE_ASSETS.
- Kode lengkapnya pada gambar berikut ini.

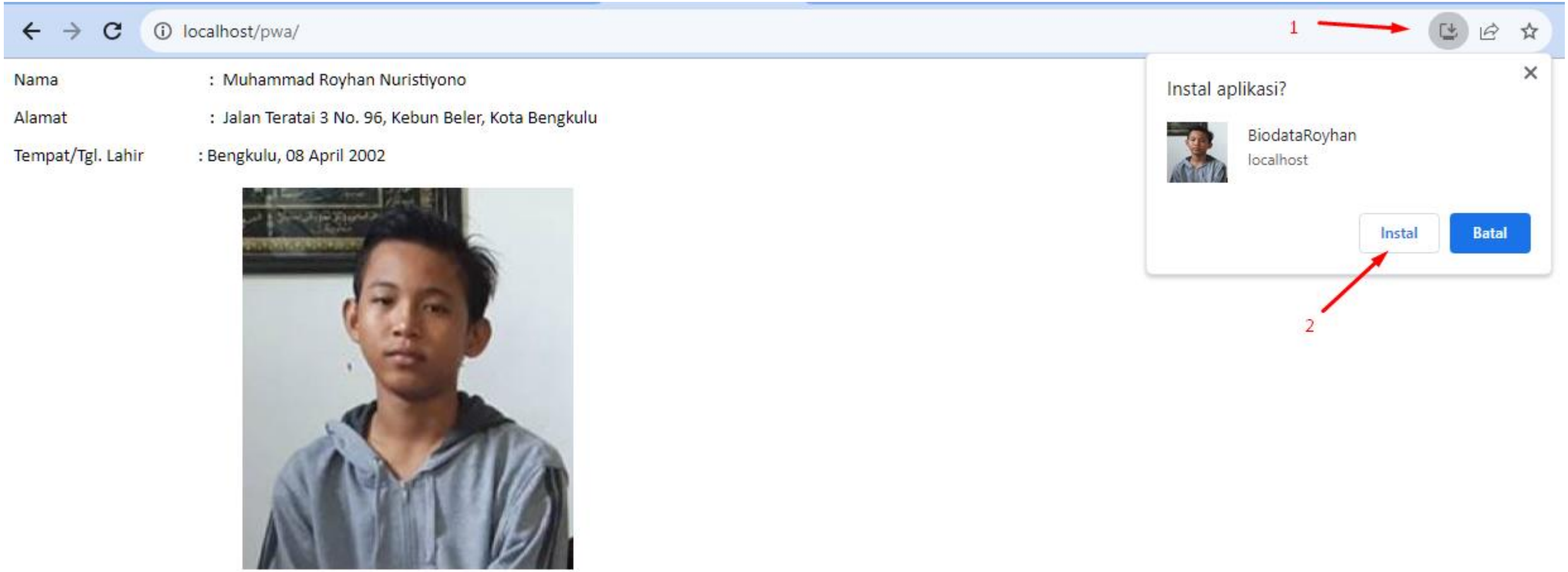
Kode Lengkap File sw.js

```
const CACHE_NAME = 'cool-cache';

// Add whichever assets you want to precache here:
const PRECACHE_ASSETS = [
    'index_files/',
    'register.js',
    'logo.png',
    'src/'
]

// Listener for the install event - precaches our assets list on service worker install
self.addEventListener('install', function(event) {
    event.waitUntil(
        caches.open(CACHE_NAME)
        .then(function(cache) {
            return cache.addAll(toCache)
        })
        .then(self.skipWaiting())
    )
})
```

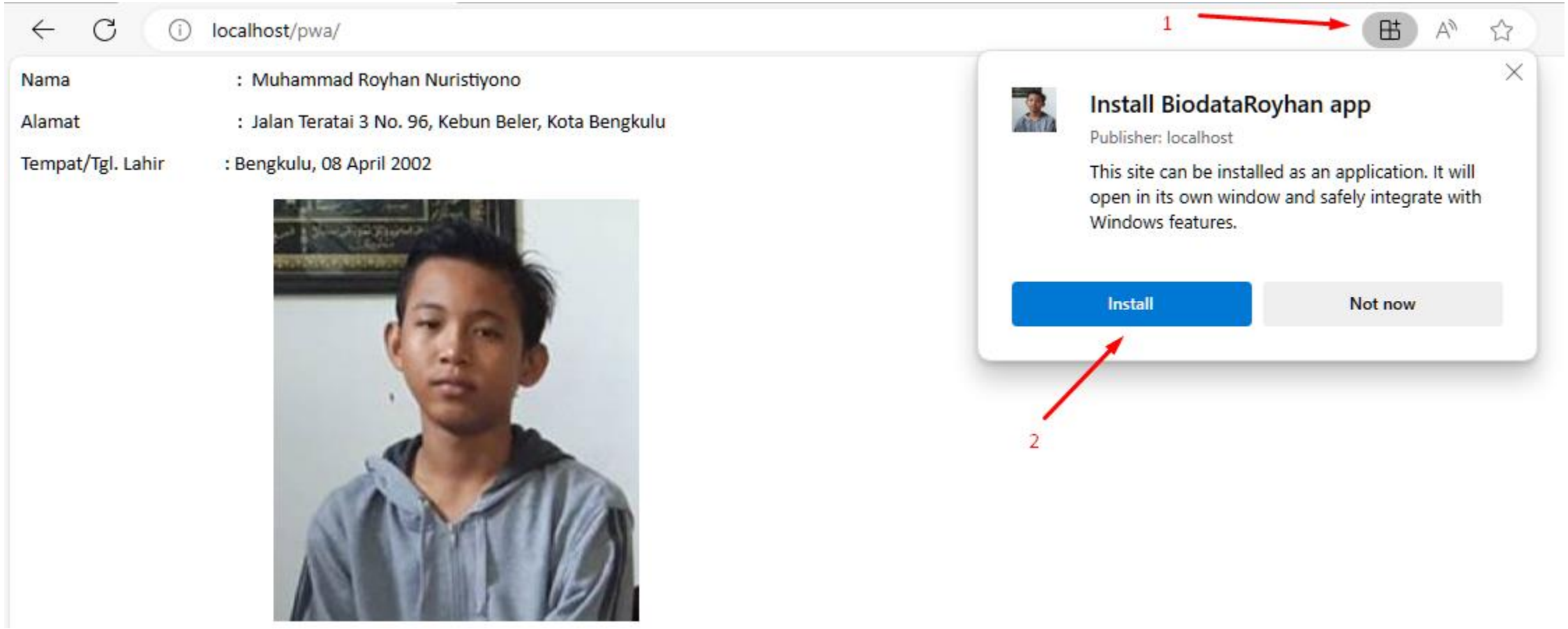
```
21
22  self.addEventListener('fetch', function(event) {
23      event.respondWith(
24          fetch(event.request)
25          .catch(() => {
26              return caches.open(CACHE_NAME)
27              .then((cache) => {
28                  return cache.match(event.request)
29              })
30          })
31      )
32  })
33
34  self.addEventListener('activate', function(event) {
35      event.waitUntil(
36          caches.keys()
37          .then((keyList) => {
38              return Promise.all(keyList.map((key) => {
39                  if (key !== CACHE_NAME) {
40                      console.log('[ServiceWorker] Hapus cache lama', key)
41                      return caches.delete(key)
42                  }
43              }))
44          })
45          .then(() => self.clients.claim())
46      )
47  })
```

10. Jalankan Aplikasi dan Install

- Buka browser Chrome atau Edge dan akses url aplikasi serta klik icon install pada address bar browser tersebut seperti pada gambar.



Tampilan Pada Browser Edge



Tampilan Akses dan Install Pada Smartphone