

FIREBASE CRUD

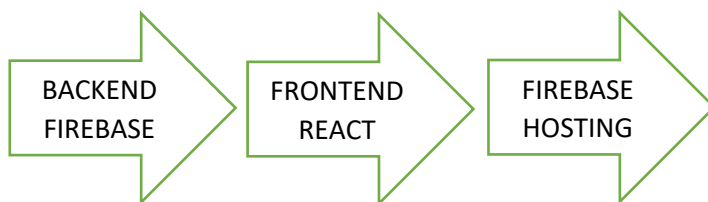
MUDAH CEPAT GRATIS

Bahasa Pemrograman : Java Script

Firebase alias BaaS (*Backend as a Service*) merupakan solusi yang ditawarkan oleh Google untuk mempercepat pekerjaan developer.

Dengan menggunakan Firebase, apps developer bisa fokus dalam mengembangkan aplikasi tanpa memberikan *effort* yang besar untuk urusan *backend*.

<https://firebase.google.com/docs/guides?hl=id>



NAMA

KEGUNAAN

FIREBASE

Nama Aplikasi Cloud Google (induk)

Authentication

Service untuk Autentikasi

Realtime Database

Database Realtime seperti mongodb

Cloud Firestore

Database Firestore lebih canggih dari Database Realtime

Storage

Service untuk menyimpan file

Hosting

Service untuk hosting (xxx.web.app)

Cloud Function

Service untuk function seperti API/Backend berbasis node.js

Jenis data

Cloud Firestore memungkinkan Anda untuk menulis berbagai jenis data di dalam dokumen, termasuk string, boolean, angka, tanggal, null, dan array serta objek bertingkat. Cloud Firestore selalu menyimpan angka sebagai nomor ganda, terlepas dari jenis angka yang Anda gunakan dalam kode.

```
var docData = {
  stringExample: "Hello world!",
  booleanExample: true,
  numberExample: 3.14159265,
  dateExample: firebase.firestore.Timestamp.fromDate(
    new Date("December 10, 1815")
  ),
  arrayExample: [5, true, "hello"],
  nullExample: null,
  objectExample: {
    a: 5,
    b: {
      nested: "foo",
    },
  },
};
db.collection("data")
  .doc("one")
  .set(docData)
  .then(() => {
    console.log("Document successfully written!");
  });
```

GITHUB

<https://github.com/edylee/fire-crud>

```
-----clone
git clone https://github.com/edylee/fire-crud.git
```

```
//fire-crud/functions
npm install
//fire-crud/coba-crud
npm install
//fire-crud/admin-crud
npm install
```

daftar isi di github setiap branch

<https://github.com/edylee/fire-crud/tree/latihan3>
<https://github.com/edylee/fire-crud/blob/latihan3/admin-crud/src/Latihan3.js>
read collection dari firestore database

<https://github.com/edycoleeee/fire-crud/tree/latihan3a>
<https://github.com/edycoleeee/fire-crud/blob/latihan3a/admin-crud/src/Latihan3.js>
read collection dan dokumen dari firestore database

<https://github.com/edycoleeee/fire-crud/tree/latihan4>
<https://github.com/edycoleeee/fire-crud/blob/latihan4/admin-crud/src/Latihan4.js>
read collection dari firestore database ditampilkan ke table

<https://github.com/edycoleeee/fire-crud/tree/latihan5>
<https://github.com/edycoleeee/fire-crud/blob/latihan5/admin-crud/src/Latihan5.js>
edit delete ke firestore tampilan statis

<https://github.com/edycoleeee/fire-crud/tree/latihan5a>
<https://github.com/edycoleeee/fire-crud/blob/latihan5a/admin-crud/src/Latihan5.js>
edit ke firestore tampilan dinamis

<https://github.com/edycoleeee/fire-crud/tree/latihan5b>
<https://github.com/edycoleeee/fire-crud/blob/latihan5b/admin-crud/src/Latihan5.js>
delete ke firestore tampilan dinamis

<https://github.com/edycoleeee/fire-crud/tree/latihan6>
<https://github.com/edycoleeee/fire-crud/blob/latihan6/admin-crud/src/Latihan6.js>
add dan delete field tambahan

<https://github.com/edycoleeee/fire-crud/tree/latihan6a>
<https://github.com/edycoleeee/fire-crud/blob/latihan6a/admin-crud/src/Latihan6.js>
add server time ke database

<https://github.com/edycoleeee/fire-crud/tree/latihan7>
<https://github.com/edycoleeee/fire-crud/blob/latihan7/admin-crud/src/Latihan7.js>
CRUD ke firestore database

MEMPERSIAPKAN BACKEND FIREBASE

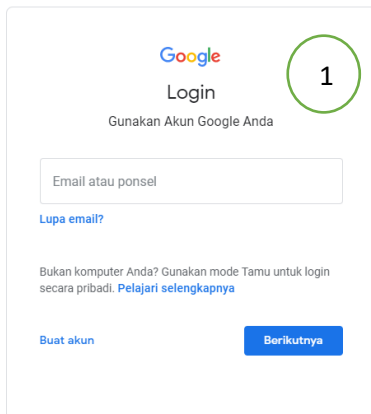
A. MENDAPATKAN FIREBASE CONFIG



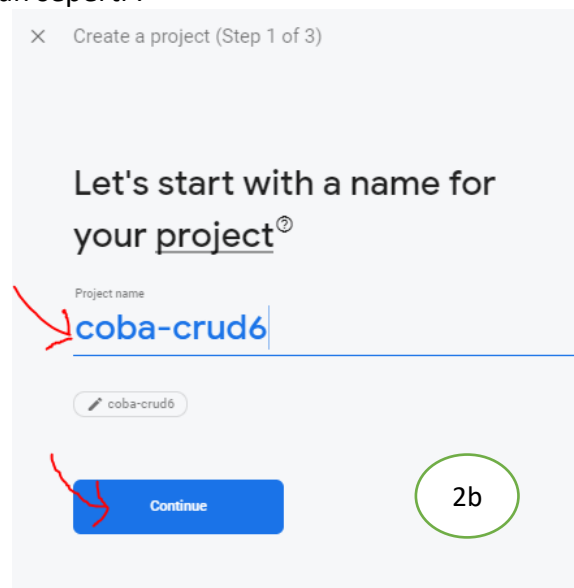
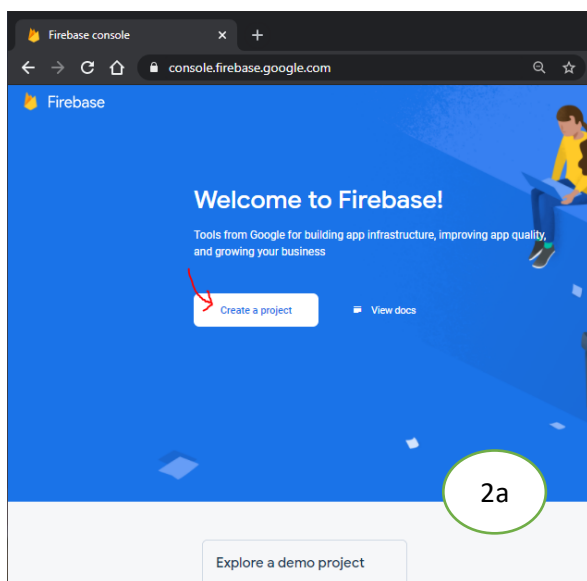
1. LOGIN FIREBASE CONSOLE DENGAN AKUN GOOGLE

2. CREATE NEW PROJECT

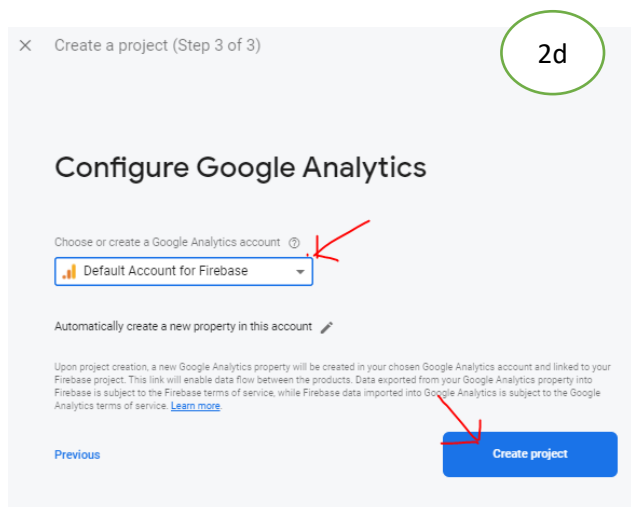
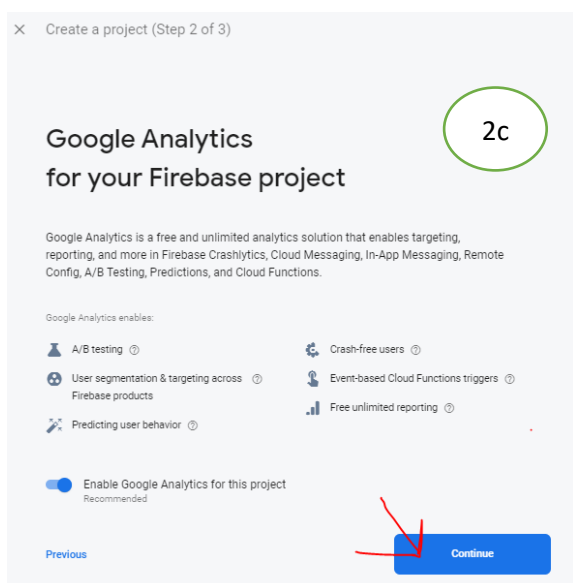
Membuka Link <https://console.firebase.google.com/>



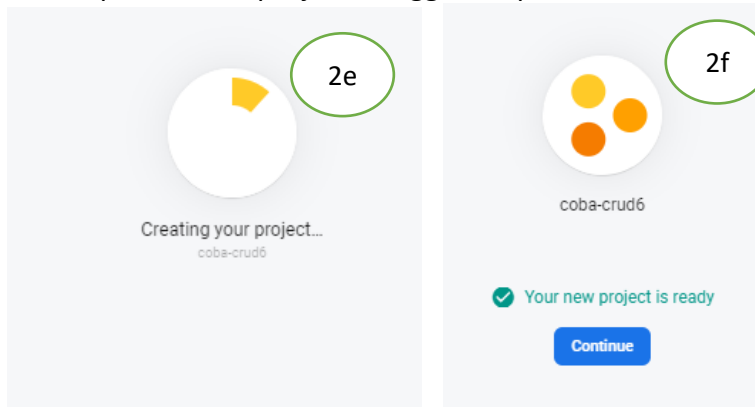
Login dengan Akun Google maka tampilan akan seperti :



Google memberikan fasilitas google analitic dengan gratis, jika ingin menggunakannya bisa seperti langkah dibawah, jika tidak maka pilih tombol disable.

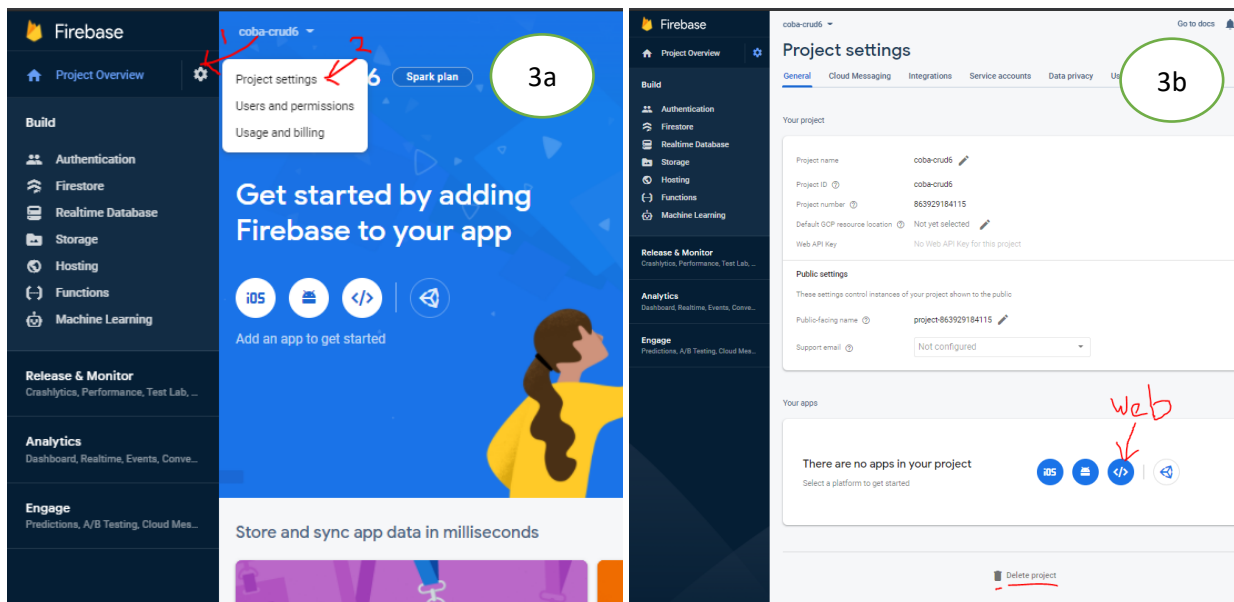


Proses pembuatan project, tunggu sampai selesai

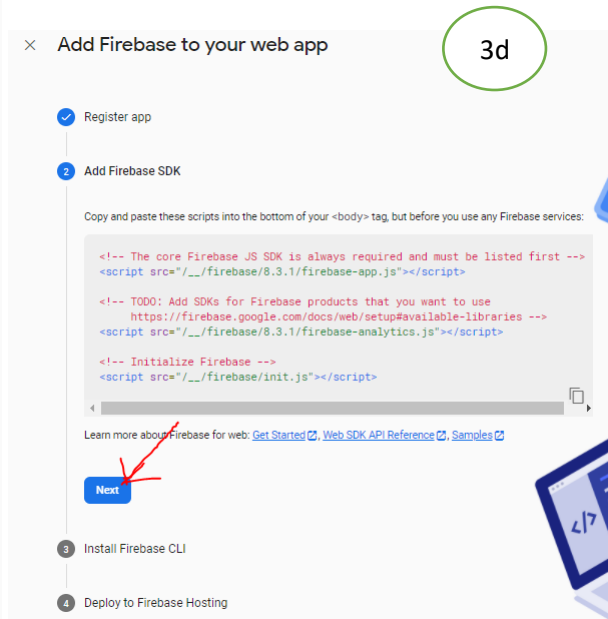
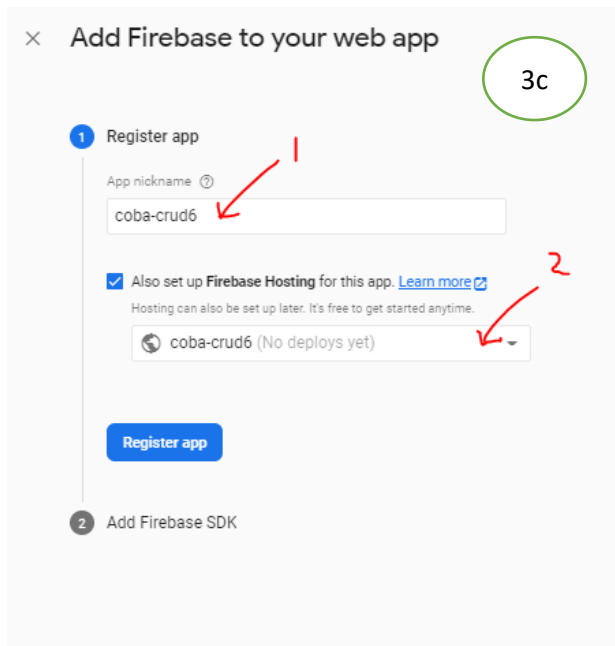


3. REGISTER APP KE DALAM PROJECT

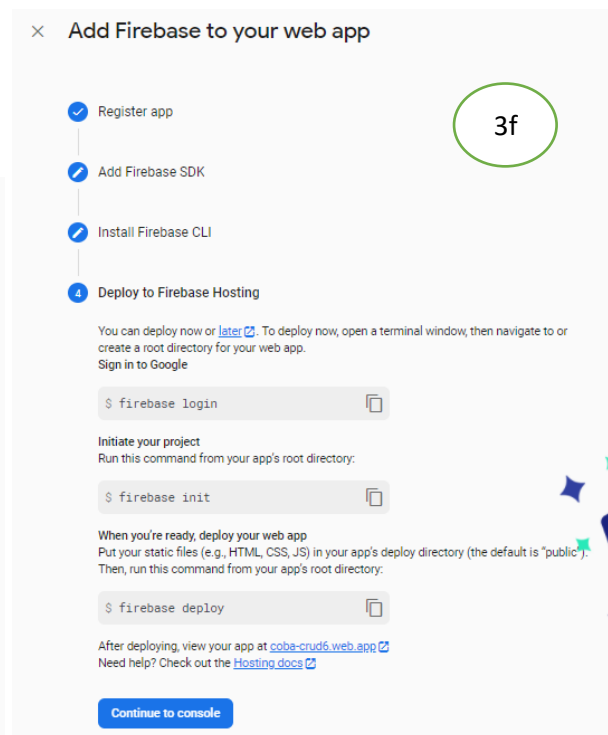
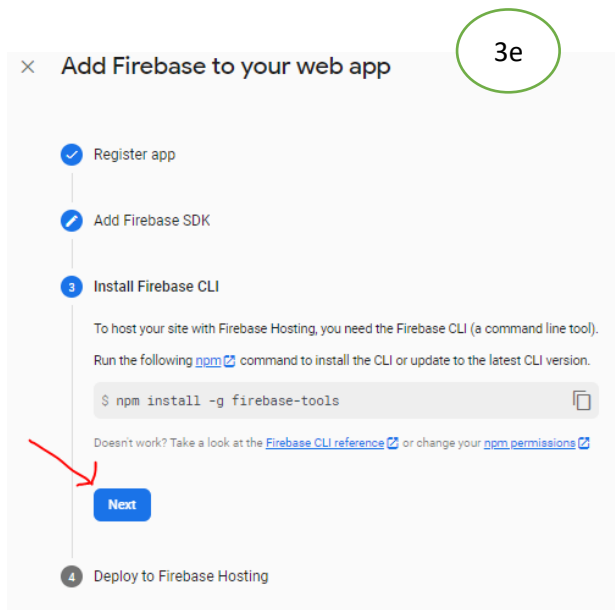
Frontend yang kita gunakan nanti adalah reactJS maka register apps kita pilih yang web



Penamaan app, secara bebas, atau boleh juga disamakan dengan nama project, bisa jadi dalam satu project terdiri banyak apps sehingga penamaan app ini sesuai dengan kebutuhan kita.

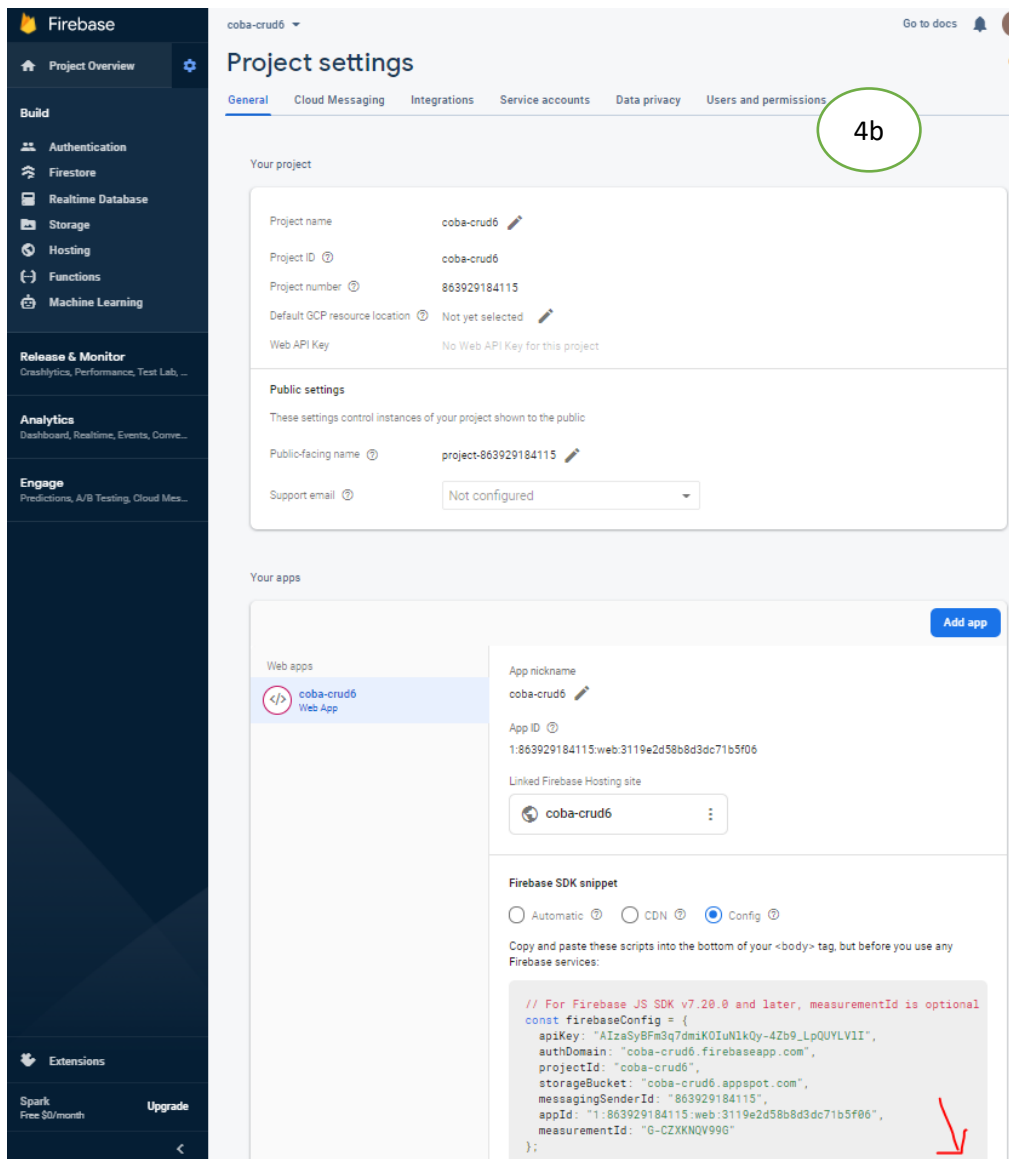
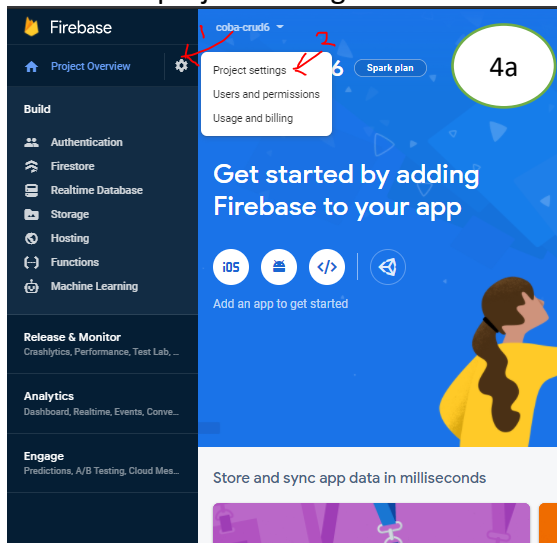


Bisa juga disaat awal ini tidak memilih hosting, hanya memanfaatkan firebase firestore database, sedangkan hosting diluar firestore. Pada tutorial kali ini kita pilih juga layanan hosting firebase.



4. MENDAPATKAN FIREBASE CONFIG

Kembali ke project setting maka akan kita dapatkan kode firebase config



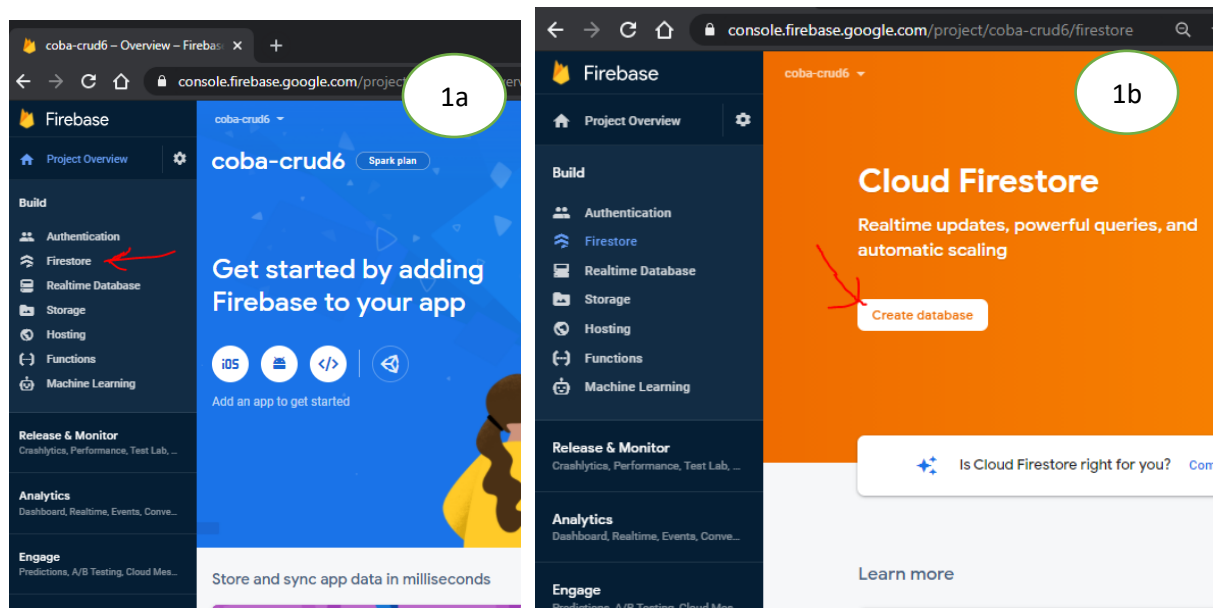
```
// For Firebase JS SDK v7.20.0 and later, measurementId is optional
const firebaseConfig = {
  apiKey: "AIzaSyBFm3q7dmikOIuNlkQy-4Zb9_LpQUYLVlI",
  authDomain: "coba-crud6.firebaseio.com",
  projectId: "coba-crud6",
  storageBucket: "coba-crud6.appspot.com",
  messagingSenderId: "863929184115",
  appId: "1:863929184115:web:3119e2d58b8d3dc71b5f06",
  measurementId: "G-CZXKNQV99G"
};
```

5. CREATE DATABASE FIRESTORE

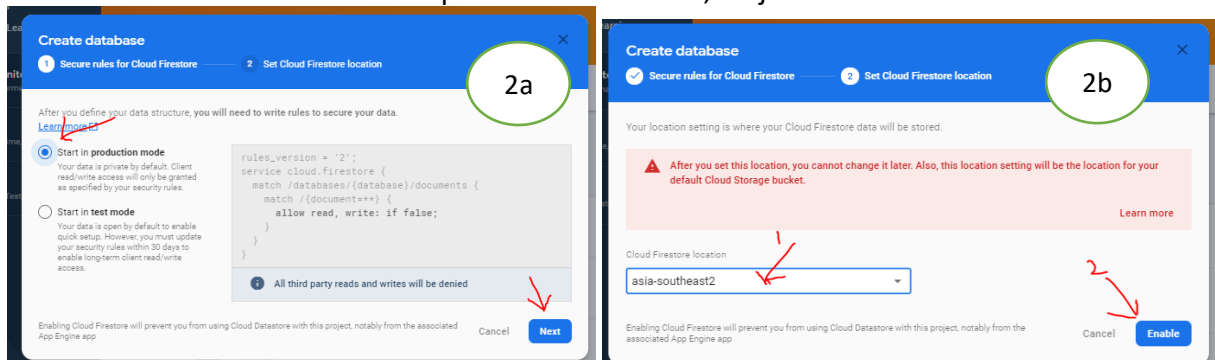
Firestore Database sejak tahun 2021 telah mengalami beberapa perubahan yang agak sedikit berbeda dari versi sebelumnya. adanya pilihan native mode dan update firestore rule melalui Firebase CLI.



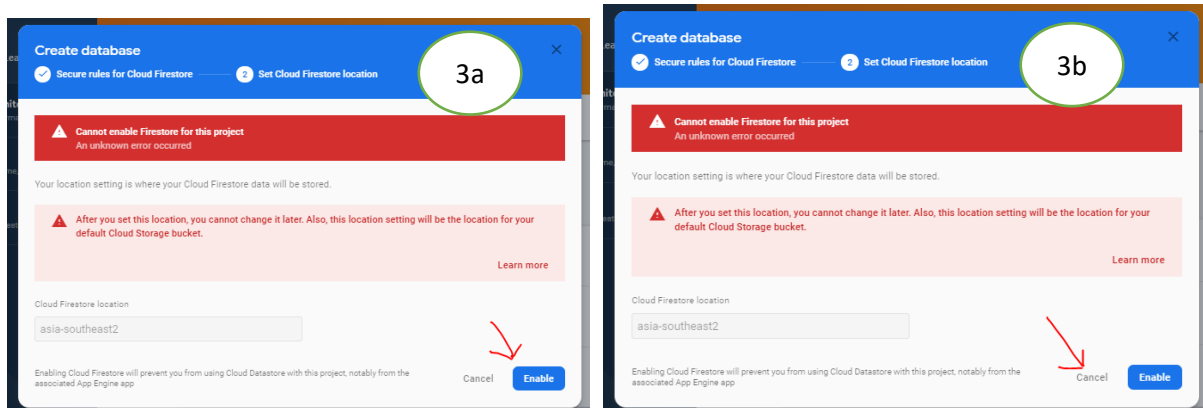
Membuat Firestore database



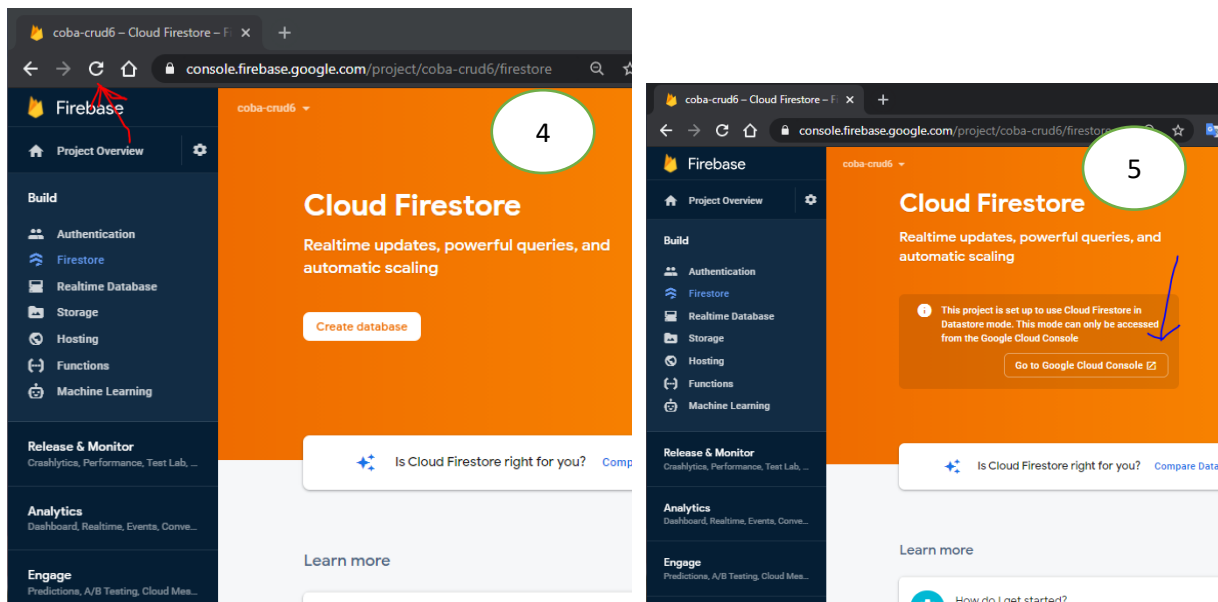
Pilih Production Mode kemudian pilih lokasi database, lanjutkan Enable



Firebase akan mengatakan tidak bisa mengaktifkan firestore ke dalam project, kemudian pilih cancel dan refresh.



KEMBALI DAN REFRESH SAMPAI MUNCUL “GOTO GOOGLE CLOUD”. jika belum ada link tersebut maka Enablekan (langkah 2a,2b) kembali sampai setelah di refreh akan muncul “GOTO GOOGLE CLOUD”.



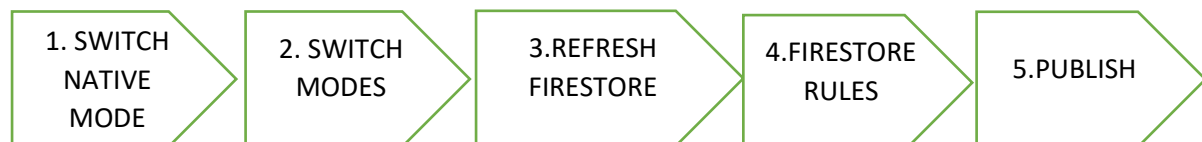
Data pilihan letak database Firebase. Pemilihan lokasi ini tidak bisa diedit memilih lokasi lain setelah pemilihan lokasi pertama kali, jadi penentuan ini hanya sekali dalam membuat sebuah project.

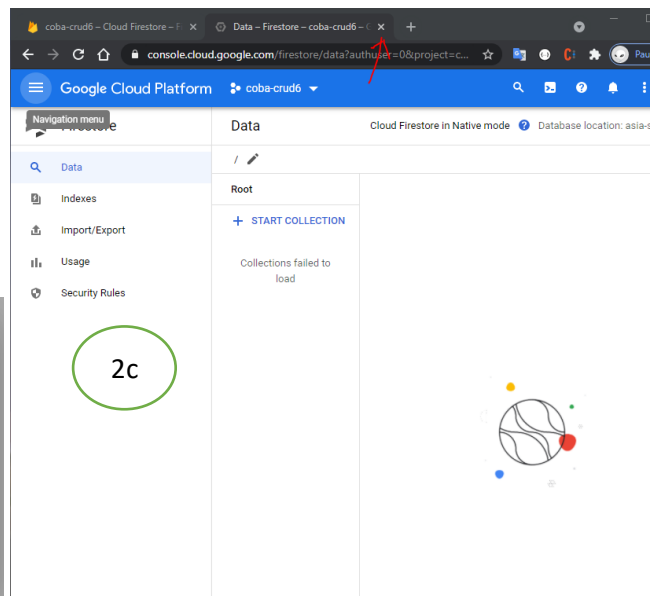
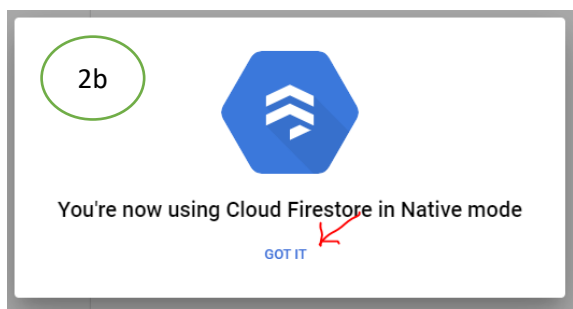
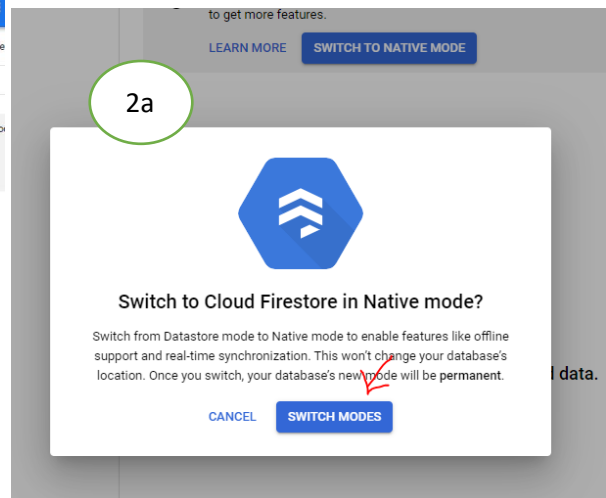
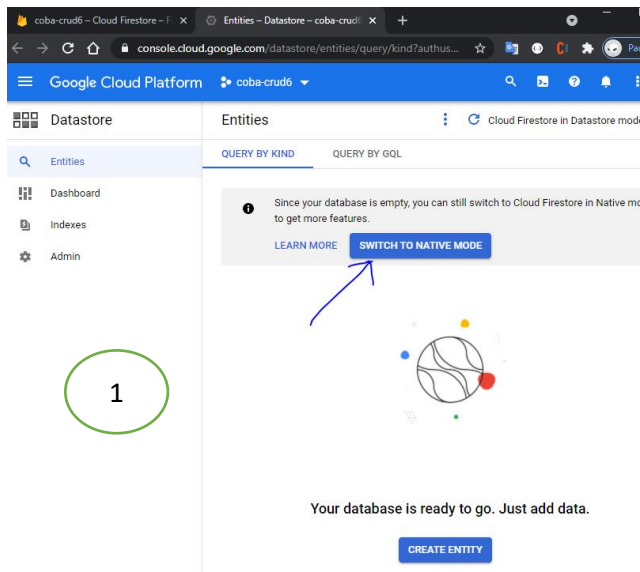
Firebase mendukung lokasi resource GCP regional berikut:

Nama Region	Deskripsi Region
Amerika Utara	
us-west2	Los Angeles
us-west3	Salt Lake City
us-west4	Las Vegas
northamerica-northeast1	Montréal
us-east1	South Carolina
us-east4	Northern Virginia
Amerika Selatan	
southamerica-east1	Sao Paulo
Eropa	
europa-west2	London
europa-west3	Frankfurt
europa-west6	Zürich
Asia	
asia-south1	Mumbai
asia-southeast2	Jakarta
asia-east2	Hong Kong
asia-northeast1	Tokyo
asia-northeast2	Osaka
asia-northeast3	Seoul
Australia	
australia-southeast1	Sydney

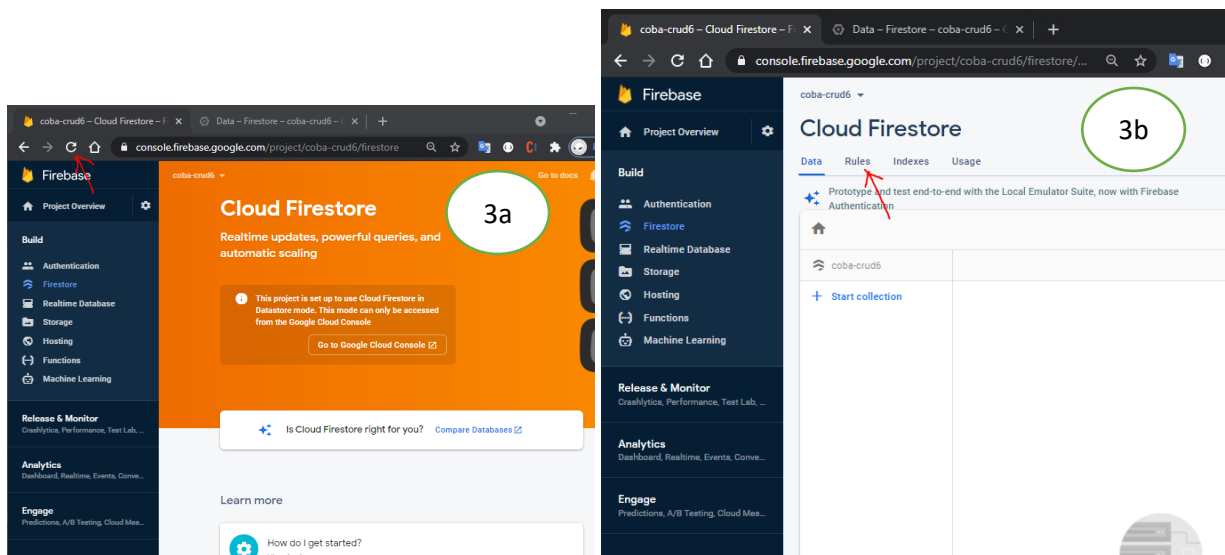
6. AKTIFKAN KE NATIVE MODE

Masuk ke dalam Google Cloud Platform dan pilih mode native mode, kemudian kembali ke firestore maka firestore database sudah enable mode

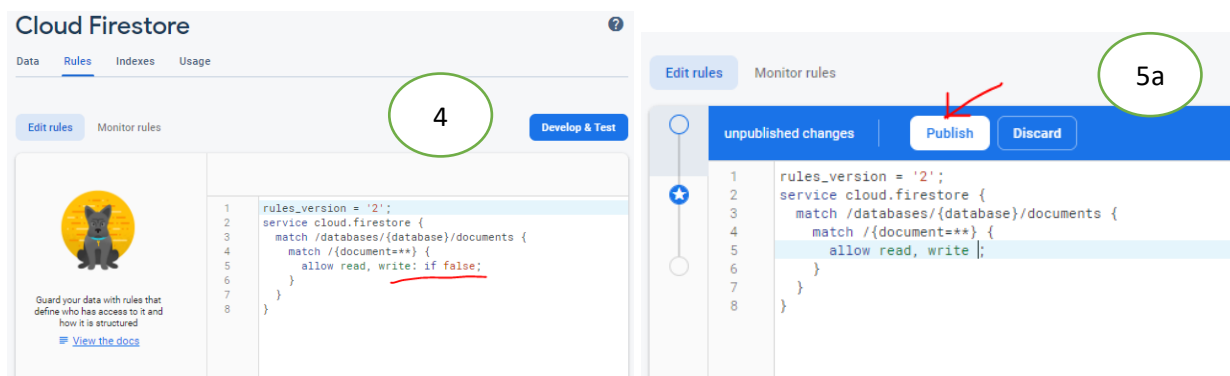




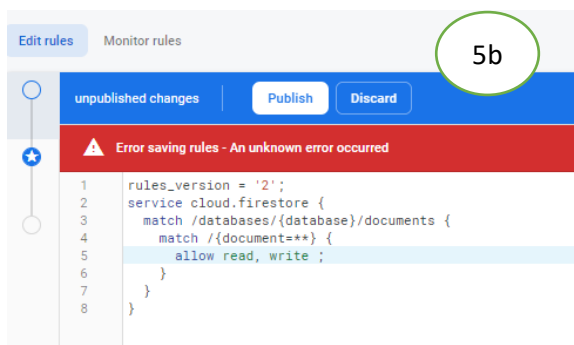
Kembali ke firebase console dan refresh page, maka tampilan firestore database akan terlihat (enable). Kemudian edit Rules Cloud Firestore untuk bisa melakukan write dari aplikasi front end.

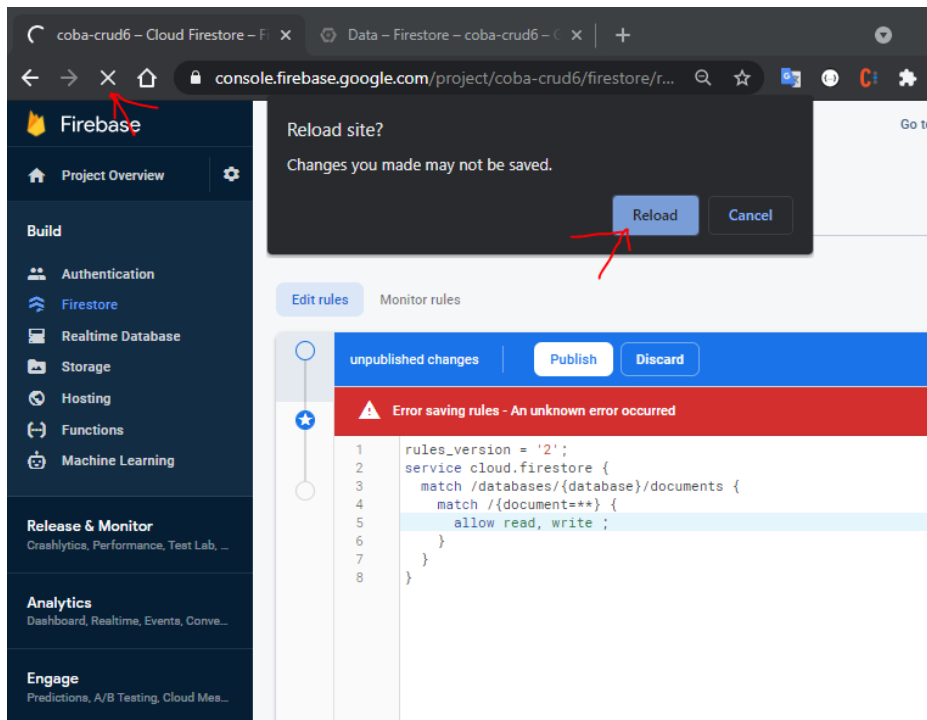


<https://firebase.google.com/docs/firestore/security/get-started>



DAN TERJADI ERROR MAKA REFRESH PAGE DAN RULE INI TIDAK BISA DIUBAH DARI SINI
 KEJADIAN INI TERJADI TAHUN 1 JANUARI 2021, UNTUK MENGUBAH RULE HARUS MELALUI
 FIREBASE CLI
 DAN TANPA MENGUBAH ITU KITA TIDAK BISA WRITE KE FIRESTORE
 RUMIT YA... MEMANG SEJAK 2021 SEMUA JADI RUMIT





Setelah maret 2021 bug ini sudah diperbaiki oleh google, dan bisa di edit serta publish langsung lewat firebase console

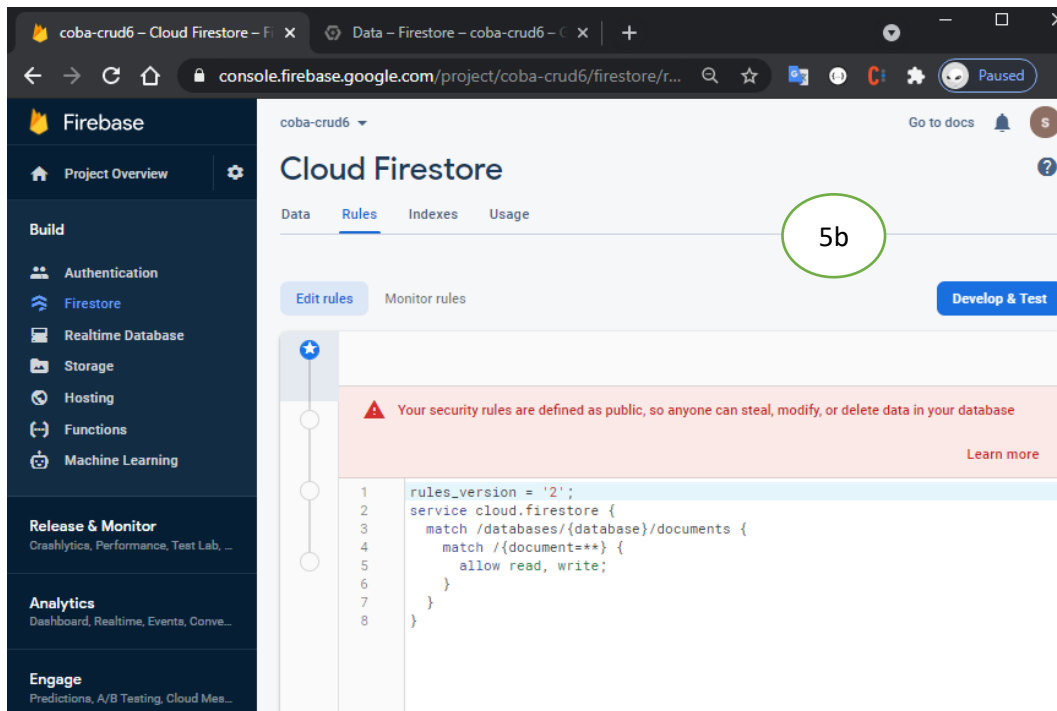
```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
  }
}
```

4a

MAKA KITA hilangkan "if false" MENJADI seperti dibawah DAN SIMPAN

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write ;
    }
  }
}
```

4b



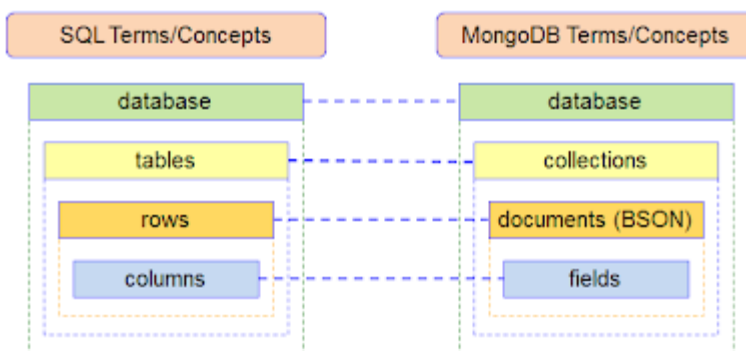
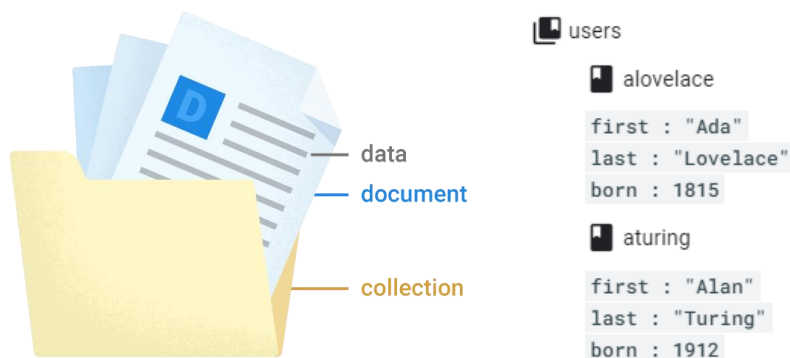
OKEH SETTING INI MEMANG BERBAHAYA KARENA SEMUA ORANG BISA AKSES KE DATABASE TANPA AUTHENTIKASI, UNTUK CRUD SAJA KITA BELAJAR DENGAN YANG MUDAH, TAHAP BERIKUTNYA NANTI KITA KASIH RULE IF AUTH AGAR DATABASE LEBIH AMAN <https://firebase.google.com/docs/firestore/security/get-started>

```
// Allow read/write access on all documents to any user signed in to the application
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.auth != null;
    }
  }
}
```

B. PEMBUATAN APLIKASI FRONTEND MENGGUNAKAN REACTJS

kita akan membuat database/ collection seperti pada contoh firebase

<https://firebase.google.com/docs/firestore/data-model>



SQL VS. NOSQL OVERSIMPLIFIED

SQL

`SELECT * FROM Customers.tbl WHERE Last_Name='Smith';`

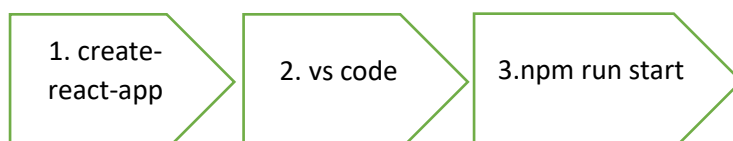
Cust_No	Last_Name	First_Name
560779	Smith	Juan
207228	Smith	George
173996	Smith	Ben
477610	Smith	Conrad

NOSQL

`Get customer.firstname,customer.lastname,customer.productID.* where Last_Name='Whitelock'`

Key	Value
746133	Firstname: George Lastname: Whitelock productID: 2012: 5
135225	Firstname: Luke Lastname: Whitelock productID: 1285: 1 1077: 5
884256	Firstname: Sam Lastname: Whitelock productID: 1442: 2

1. MENJALANKAN REACT JS



Panduan instalasi

<https://youtu.be/cFa3gPEOdRk>

yang harus diinstal adalah nodejs dan vscode

<https://nodejs.org/en/download/>
<https://code.visualstudio.com/>

masuk ke CMD

>npx create-react-app frontend

```
npm
Project Console: https://console.firebase.google.com/project/coba-crud6/over
D:\REACT\firebase-crud>npx create-react-app frontend
Creating a new React app in D:\REACT\firebase-crud\frontend.
Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts with cra-template...
[...]\ fetchMetadata: sill resolveWithNewModule estraverse@5
```

```
Command Prompt
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.
y"} (current: {"os":"win32","arch":"x64"})
removed 1 package and audited 1950 packages in 10.009s
133 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities

Created git commit.

Success! Created frontend at D:\REACT\firebase-crud\frontend
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

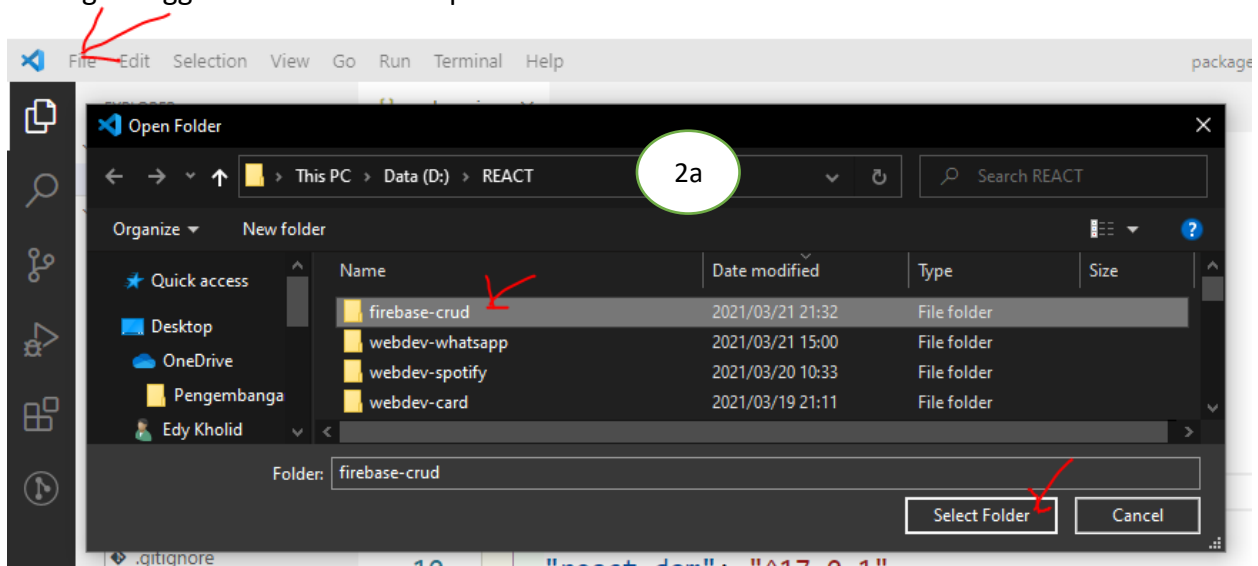
  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

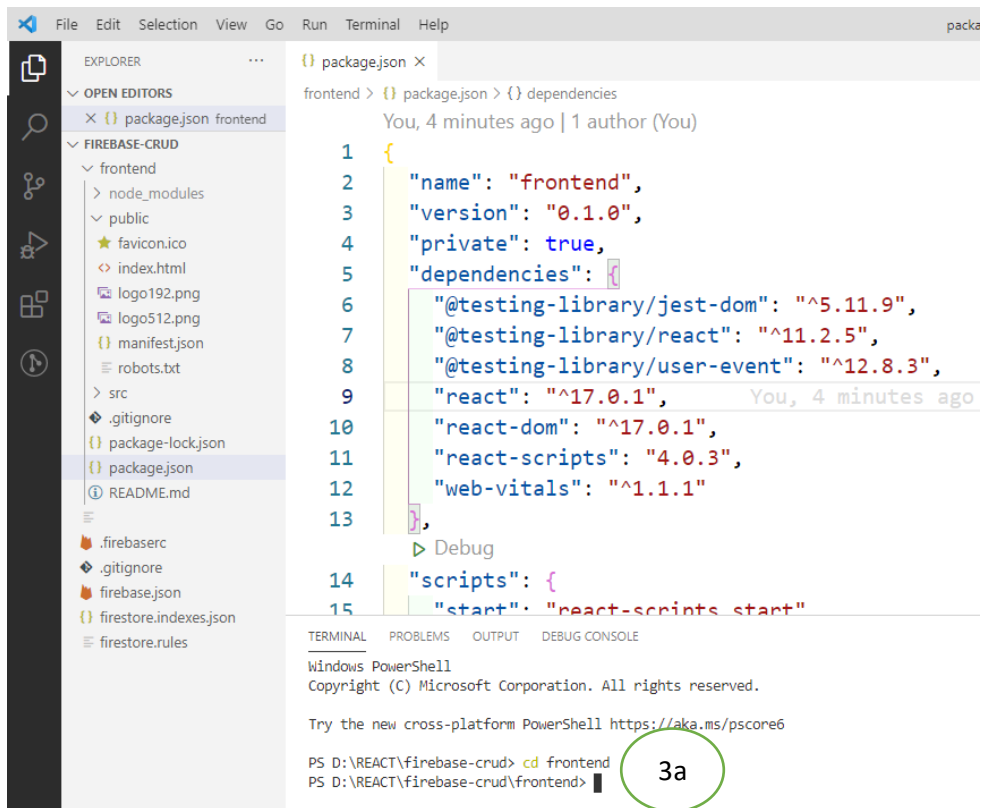
We suggest that you begin by typing:

  cd frontend
  npm start

Happy hacking!
D:\REACT\firebase-crud>
```

editing menggunakan vscode => open folder

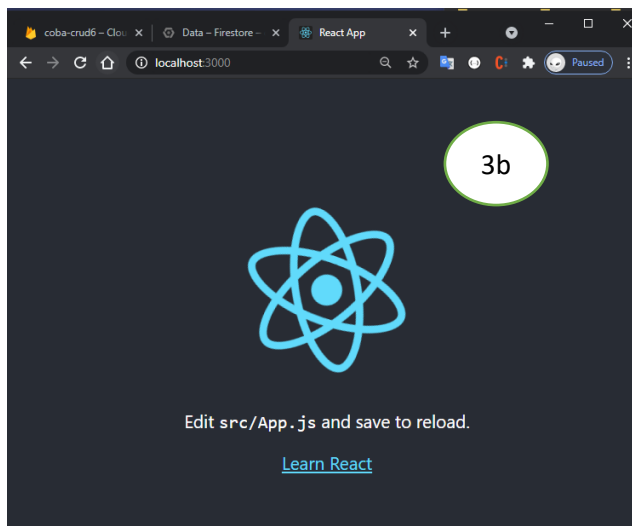




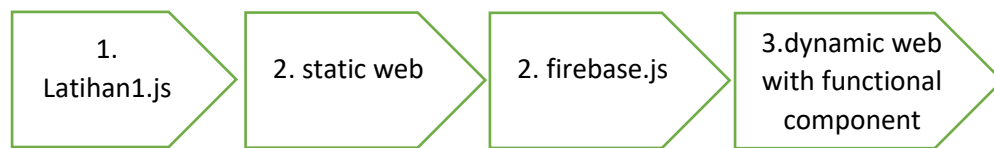
```

cd frontend
npm install --save firebase
npm run start

```



2. WRITE DATA KE FIRESTORE



2.1. Membuat File Latihan1.js

```
//src/Latihan1.js
import React from "react";

function Latihan1() {
  return (
    <div>
      <h3>Latihan1</h3>
    </div>
  );
}

export default Latihan1;
```

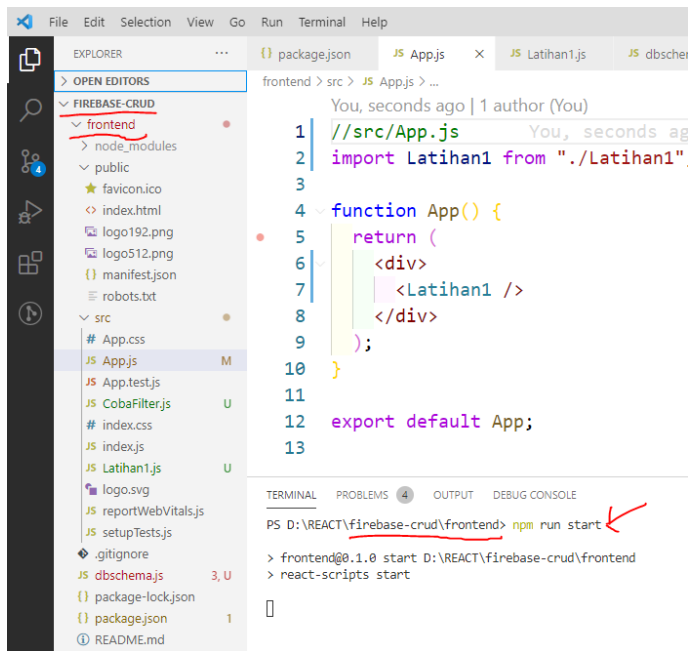
Loading Latihan 1 ke App.js

```
//src/App.js
import Latihan1 from "../Latihan1";

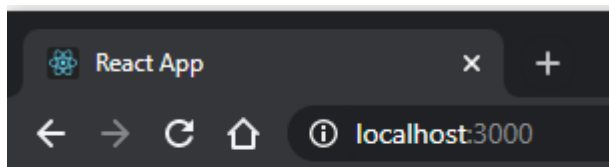
function App() {
  return (
    <div>
      <Latihan1 />
    </div>
  );
}

export default App;
```

Jalankan React dengan npm run start



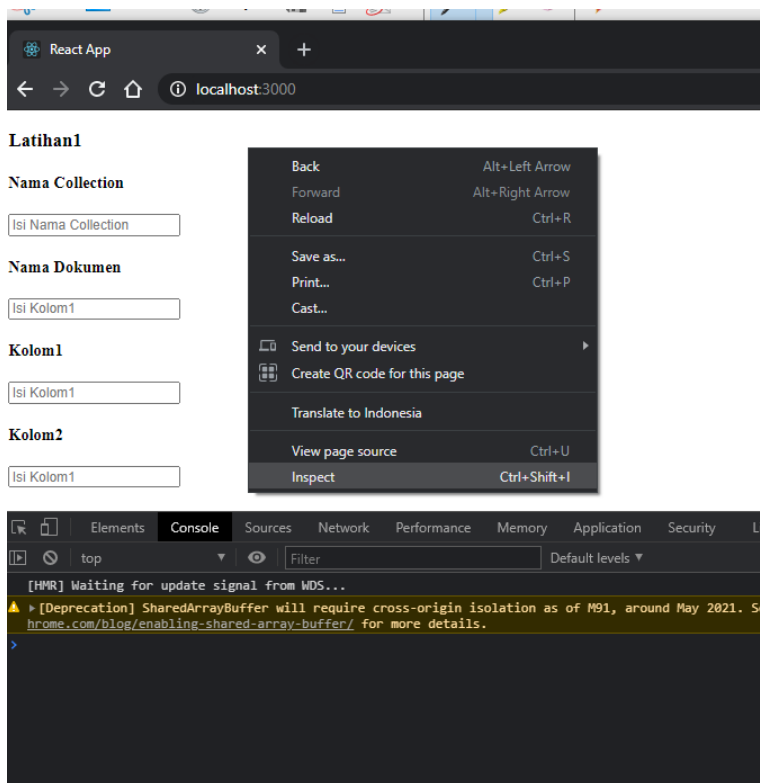
maka tampilan di browser chrome akan seperti ini, <http://localhost:3000/>



Latihan1

2.2. Static Web Latihan1.js

Klik kanan mouse dan pilih inspect pada halaman web chrome, kemudian pilih console nantinya apa yang kita ketik di isian akan tampil kedalam console



<https://github.com/edylee/fire-crud/blob/latihan1/admin-crud/src/Latihan1.js>

```
//src/Latihan1.js
import React from "react";

function Latihan1() {
  return (
    <div>
      <h3>Latihan1</h3>
      <h4>Nama Collection</h4>
      <input
        type="text"
        name="nmCollection"
        placeholder="Isi Nama Collection"
        onChange={(e) => console.log(e.target.value)}
      />
      <h4>Nama Dokumen</h4>
      <input
        type="text"
        name="nmDokumen"
        placeholder="Isi Kolom1"
        onChange={(e) => console.log(e.target.value)}
      />
      <h4>Kolom1</h4>
      <input
        type="text"
        name="kolom1"
      />
    </div>
  );
}
```

```

        placeholder="Isi Kolom1"
        onChange={(e) => console.log(e.target.value)}
      />
      <h4>Kolom2</h4>
      <input
        type="text"
        name="kolom2"
        placeholder="Isi Kolom1"
        onChange={(e) => console.log(e.target.value)}
      />
    </div>
  );
}

export default Latihan1;

```

2.2. firebase.js

Masukkan firebase config ke dalam firebase.js

```

//src/firebase.js
//import * as firebase from "firebase/app"; //before firebase v 8.00
import firebase from "firebase/app";
import "firebase/auth";
import "firebase/firestore";
import "firebase/storage";

const app = firebase.initializeApp({
  apiKey: "AIzaSyBFm3q7dmikOIuNlkQy-4Zb9_LpQUYLVlI",
  authDomain: "coba-crud6.firebaseio.com",
  projectId: "coba-crud6",
  storageBucket: "coba-crud6.appspot.com",
  messagingSenderId: "863929184115",
  appId: "1:863929184115:web:3119e2d58b8d3dc71b5f06",
});

export const auth = app.auth();
export const db = app.firestore();
export const storage = firebase.storage();
export const Firebase = firebase;

```

2.3a. Dynamic web dengan fuctional component

Masukkan firebase config ke dalam firebase.js

Menggunakan useState untuk menangkap state perubahan text input

```

//src/Latihan1.js
import React, { useState } from "react";
//import firebase
import { db } from "../firebase";

```

```

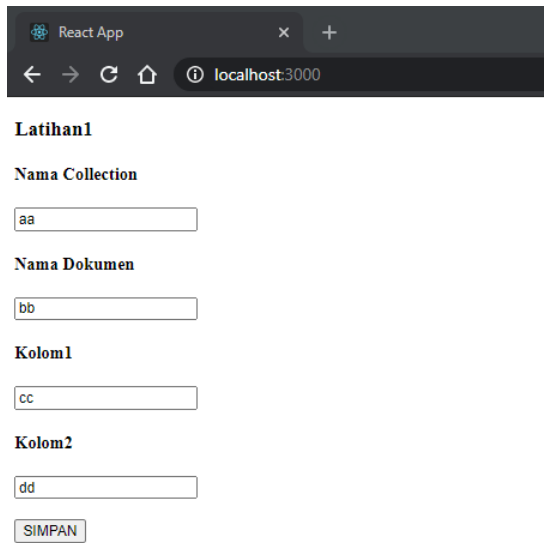
function Latihan1() {
  //state
  const [nmCollection, setNmCollection] = useState("");
  const [nmDokumen, setNmDokumen] = useState("");
  const [kolom1, setKolom1] = useState("");
  const [kolom2, setKolom2] = useState("");

  //button fuction
  function onSimpan() {
    console.log(nmCollection, nmDokumen, kolom1, kolom2);
  }

  return (
    <div>
      <h3>Latihan1</h3>
      <h4>Nama Collection</h4>
      <input
        type="text"
        name="nmCollection"
        placeholder="Isi Nama Collection"
        onChange={(e) => setNmCollection(e.target.value)}
      />
      <h4>Nama Dokumen</h4>
      <input
        type="text"
        name="nmDokumen"
        placeholder="Isi Kolom1"
        onChange={(e) => setNmDokumen(e.target.value)}
      />
      <h4>Kolom1</h4>
      <input
        type="text"
        name="kolom1"
        placeholder="Isi Kolom1"
        onChange={(e) => setKolom1(e.target.value)}
      />
      <h4>Kolom2</h4>
      <input
        type="text"
        name="kolom2"
        placeholder="Isi Kolom2"
        onChange={(e) => setKolom2(e.target.value)}
      />
      <br></br>
      <br></br>
      <button onClick={onSimpan}>SIMPAN</button>
    </div>
  );
}

```

```
export default Latihan1;
```



React App

localhost:3000

Latihan1

Nama Collection

aa

Nama Dokumen

bb

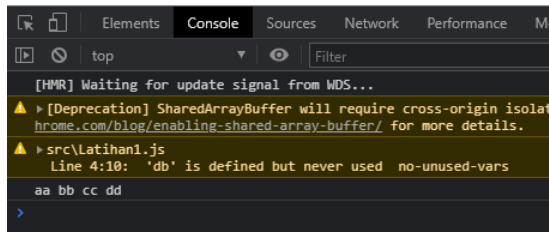
Kolom1

cc

Kolom2

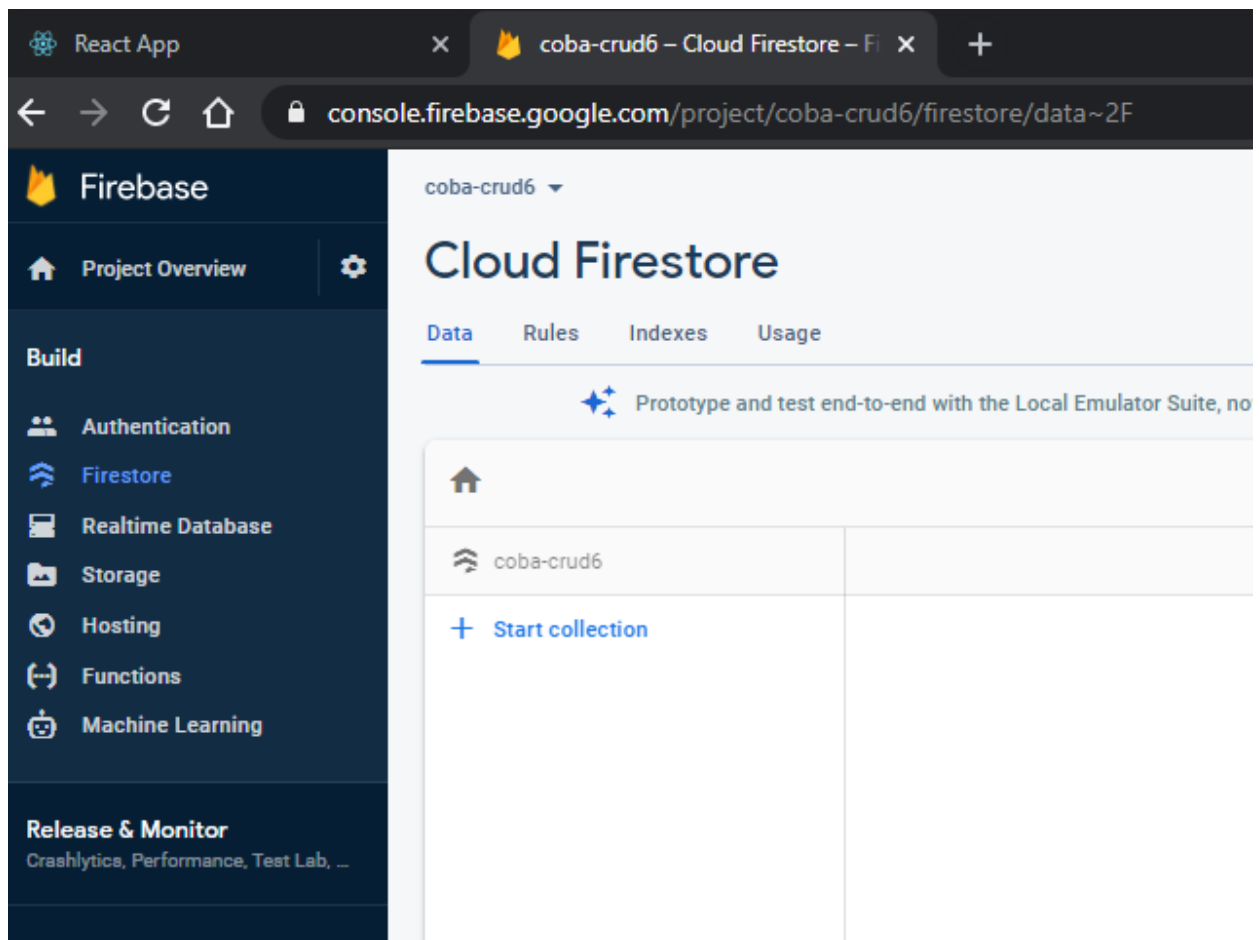
dd

SIMPAN



2.3b. Lihat database di firestore

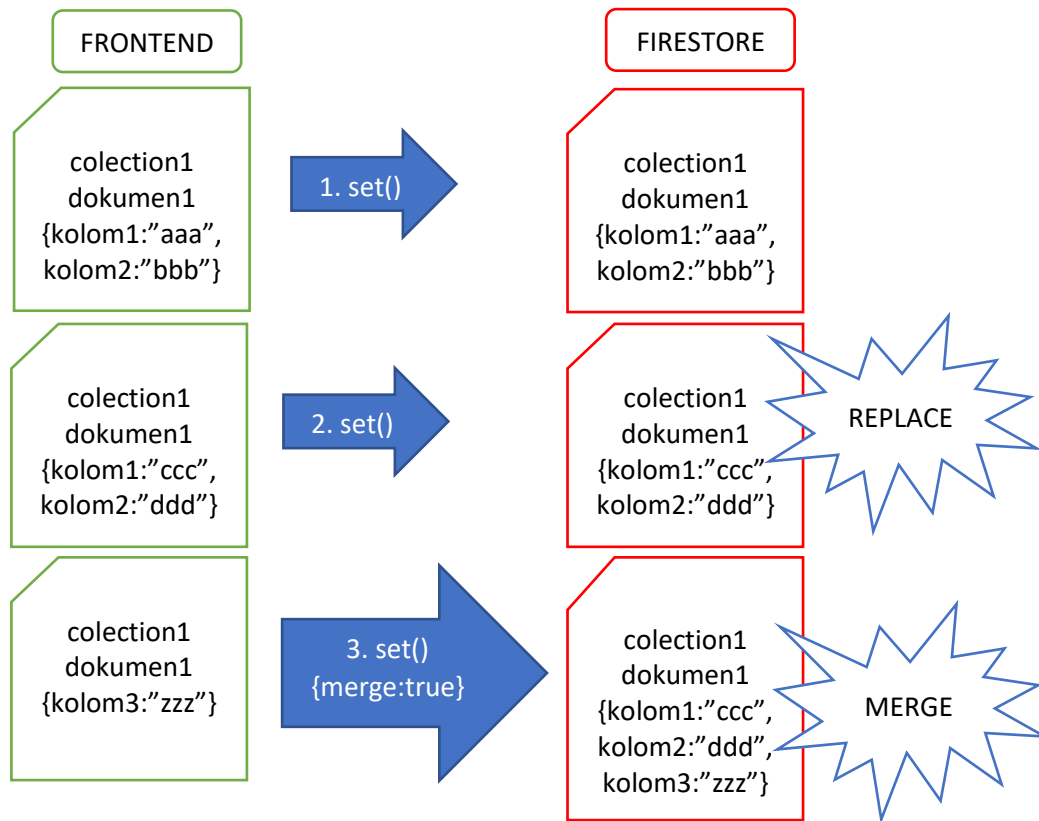
data collection masih kosong



2.3c. Write Collection dengan ID Dokumen tertentu => SET

dokumentasi mengenai cara menambahkan data kedalam cloud firestore bisa dilihat dalam link: <https://firebase.google.com/docs/firestore/manage-data/add-data?authuser=0>

Menetapkan dokumen



Untuk membuat atau menimpa sebuah dokumen, gunakan metode `set()`:

```
// Add a new document in collection "cities"
db.collection("cities").doc("LA").set({
  name: "Los Angeles",
  state: "CA",
  country: "USA"
})
.then(() => {
  console.log("Document successfully written!");
})
.catch((error) => {
  console.error("Error writing document: ", error);
});
```

Jika belum ada, dokumen itu akan dibuat. Jika dokumen sudah ada, isi kontennya akan ditimpa dengan data yang baru disediakan, kecuali jika Anda menentukan bahwa data tersebut harus digabungkan ke dalam dokumen yang ada, seperti berikut ini:

```

var cityRef = db.collection("cities").doc("BJ");

var setWithMerge = cityRef.set(
  {
    capital: true,
  },
  { merge: true }
);

```

Jika Anda tidak yakin apakah dokumen itu ada, teruskan opsi untuk menggabungkan data baru dengan dokumen yang ada agar tidak menimpa keseluruhan dokumen.

maka kita butuhkan nama collection, document dan isi datanya. Nama setiap dokumen dalam sebuah collection harus bersifat unqi, seperti penamaan table pada database SQL. Bisa juga kita lakukan write dengan tanpa nama dokumen maka secara otomatis firebase akan menambahkan unqi ID pada nama dokumen tersebut.

<https://github.com/edylee/fire-crud/blob/latihan1a/admin-crud/src/Latihan1.js>

```

//src/Latihan1.js
import React, { useState } from "react";
//import firebase
import { db } from "../firebase";

function Latihan1() {
  //state
  const [nmCollection, setNmCollection] = useState("");
  const [nmDokumen, setNmDokumen] = useState("");
  const [kolom1, setKolom1] = useState("");
  const [kolom2, setKolom2] = useState("");
  //state untuk data dari hasil pembacaan

  //button simpan fuction
  function onSimpanDokDgId() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    ;
    if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
    console.log(nmCollection, nmDokumen, kolom1, kolom2);
    db.collection(nmCollection)
      .doc(nmDokumen)
      .set({
        kolom1: kolom1,
        kolom2: kolom2,
      })
      .then(() => {
        console.log("Document successfully written!");
      })
      .catch((error) => {
        console.error("Error writing document: ", error);
      });
  }
}

```

```

    });
  }

  return (
    <div>
      <h3>Latihan1</h3>
      <h4>Nama Collection</h4>
      <input
        type="text"
        name="nmCollection"
        placeholder="Isi Nama Collection"
        onChange={(e) => setNmCollection(e.target.value)}
      />
      <h4>Nama Dokumen</h4>
      <input
        type="text"
        name="nmDokumen"
        placeholder="Isi Kolom1"
        onChange={(e) => setNmDokumen(e.target.value)}
      />
      <h4>Kolom1</h4>
      <input
        type="text"
        name="kolom1"
        placeholder="Isi Kolom1"
        onChange={(e) => setKolom1(e.target.value)}
      />
      <h4>Kolom2</h4>
      <input
        type="text"
        name="kolom2"
        placeholder="Isi Kolom2"
        onChange={(e) => setKolom2(e.target.value)}
      />
      <br></br>
      <br></br>
      <button onClick={onSimpanDokDgId}>SIMPAN DOK DG ID</button>
    </div>
  );
}

export default Latihan1;

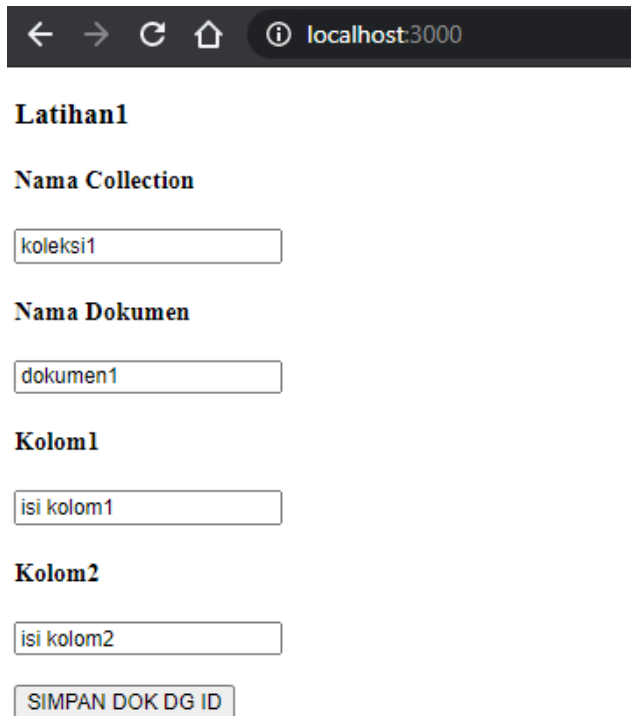
```

Data yang dikirim

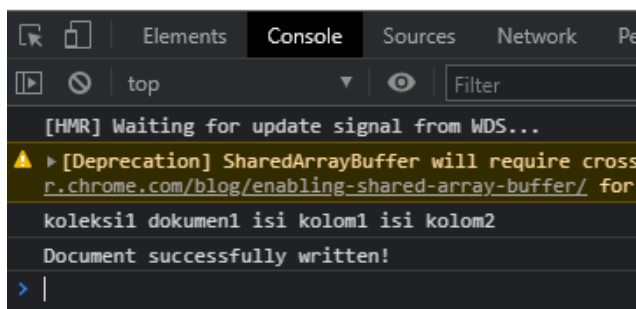
Nama Collection Koleksi1

Nama Dokumen Dokumen1

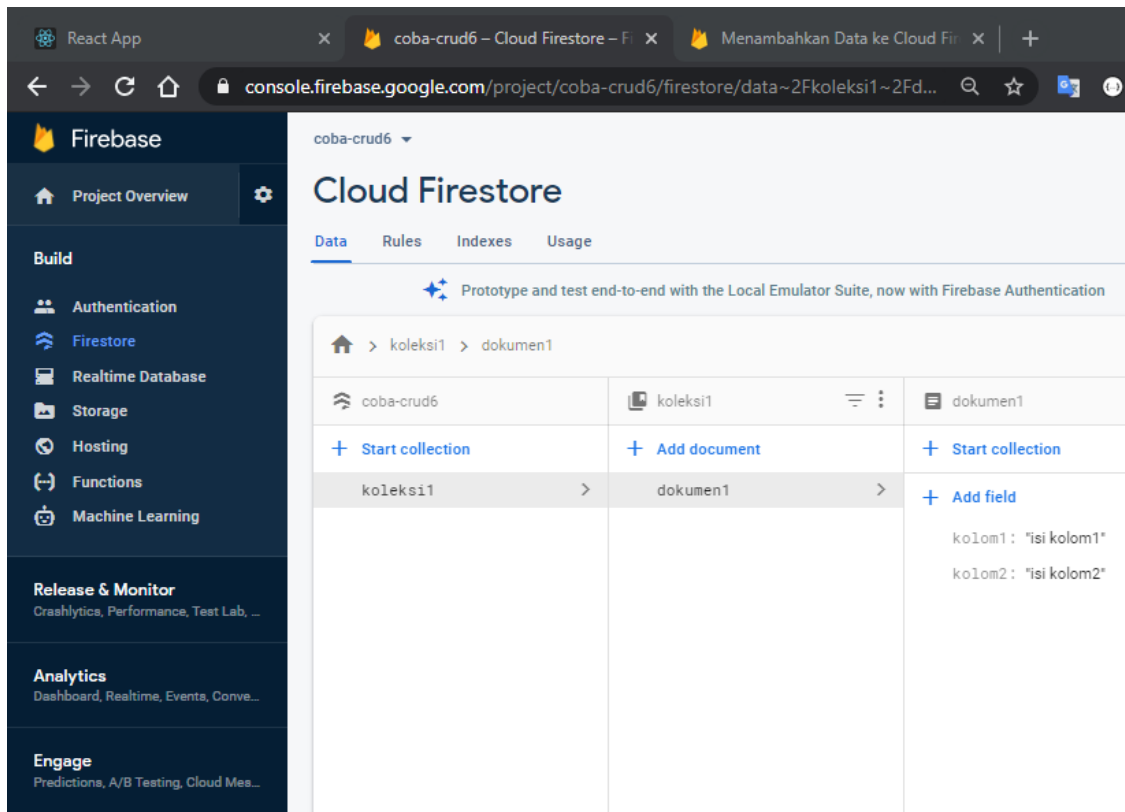
Kolom1	Isi kolom1
Kolom2	Isi kolom2



The screenshot shows a web browser window with the address bar displaying 'localhost:3000'. The page content includes a heading 'Latihan1' followed by a section 'Nama Collection' with a text input field containing 'koleksi1'. Below this is a section 'Nama Dokumen' with a text input field containing 'dokumen1'. Then there is a section 'Kolom1' with a text input field containing 'isi kolom1', and a section 'Kolom2' with a text input field containing 'isi kolom2'. At the bottom of the form is a button labeled 'SIMPAN DOK DG ID'.



dan kemudian kita lihat didalam collection firestore, refresh page dahulu jika belum muncul datanya.

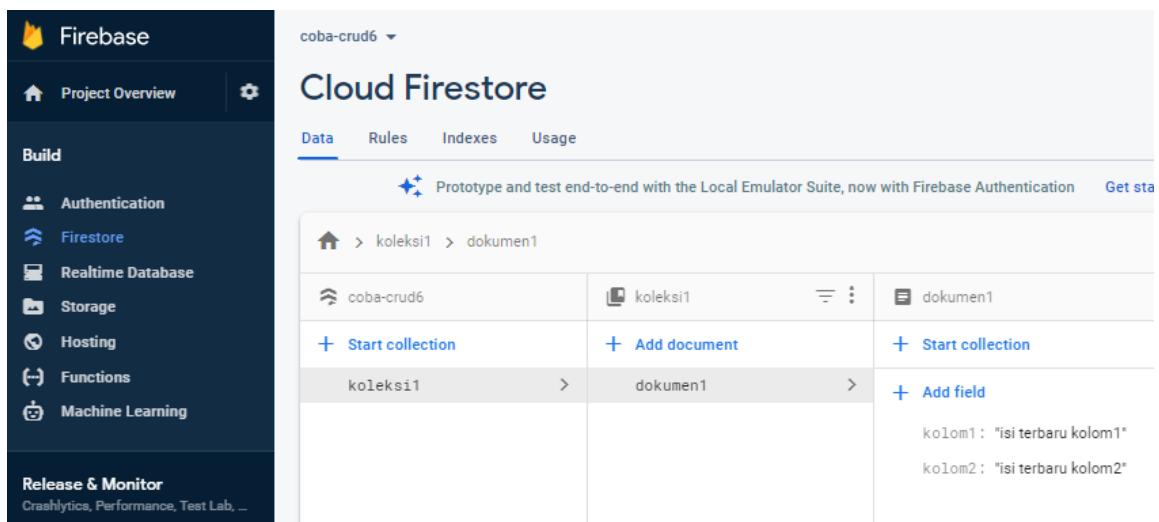


Bagaimana jika kita klik simpan lagi dengan isi data yg sama, maka dokumen tersebut akan secara otomatis mereplace dokumen yang ada.

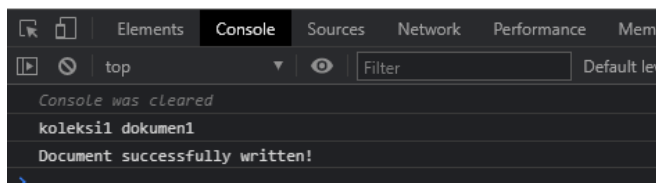
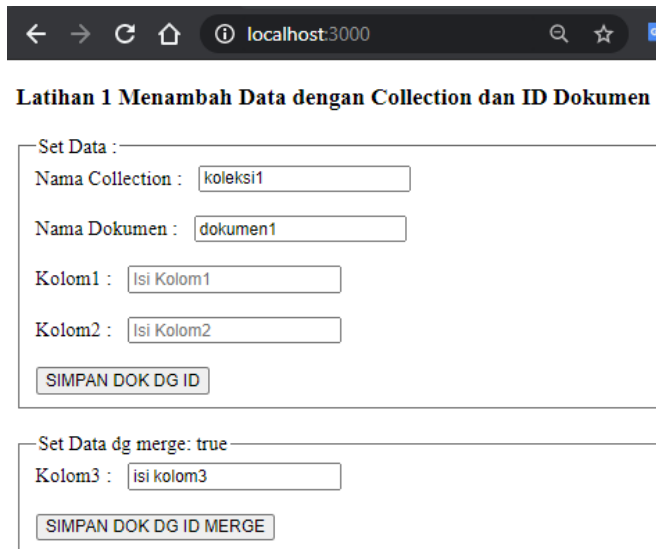
Contoh kita kirim ulang data dibawah, dengan nama koleksi dan dokumen yang sama tetapi beda isi kolomnya :

Data yang dikirim

Nama Collection	Koleksi1
Nama Dokumen	Dokumen1
Kolom1	Isi terbaru kolom1
Kolom2	Isi terbaru kolom2



dokumen tersebut akan mereplace isi yang sudah ada.
untuk menambahkan field lagi maka kita gunakan perintah merge. disini kita akan menambahkan kolom ke 3 kedalam koleksi dan dokumen tanpa mengubah isi kolom sebelumnya



<https://github.com/edycoleee/fire-crud/blob/latihan1b/admin-crud/src/Latihan1.js>

```
//src/Latihan1.js
import React, { useState } from "react";
//import firebase
import { db } from "../firebase";

function Latihan1() {
  //state
  const [nmCollection, setNmCollection] = useState("");
  const [nmDokumen, setNmDokumen] = useState("");
  const [kolom1, setKolom1] = useState("");
  const [kolom2, setKolom2] = useState("");
  const [kolom3, setKolom3] = useState("");
  //state untuk data dari hasil pembacaan

  //button simpan fuction
  function onSimpanDokDgId() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    ;
    if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
```

```

console.log(nmCollection, nmDokumen, kolom1, kolom2);
db.collection(nmCollection)
  .doc(nmDokumen)
  .set({
    kolom1: kolom1,
    kolom2: kolom2,
  })
  .then(() => {
    console.log("Document successfully written!");
  })
  .catch((error) => {
    console.error("Error writing document: ", error);
  });
}

function onSimpanDokDgIdMerge() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
;
  if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
  console.log(nmCollection, nmDokumen, kolom1, kolom2);
  db.collection(nmCollection)
    .doc(nmDokumen)
    .set(
      {
        kolom3: kolom3,
      },
      { merge: true }
    )
    .then(() => {
      console.log("Document successfully written!");
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}
return (
  <div>
    <h3>Latihan 1 Menambah Data dengan Collection dan ID Dokumen</h3>
  >
  <fieldset>
    <legend>Set Data </legend>
    <label htmlFor="nmCollection">Nama Collection </label>
    <input
      style={{ marginLeft: "1em" }}
      type="text"
      name="nmCollection"
      placeholder="Isi Nama Collection"
      onChange={(e) => setNmCollection(e.target.value)}
    />
  </fieldset>
)

```

```

<br></br>
<br></br>
<label htmlFor="nmDokumen">Nama Dokumen :</label>
<input
  style={{ marginLeft: "1em" }}
  type="text"
  name="nmDokumen"
  placeholder="Isi Kolom1"
  onChange={(e) => setNmDokumen(e.target.value)}
/>
<br></br>
<br></br>
<label htmlFor="kolom1">Kolom1 :</label>
<input
  style={{ marginLeft: "1em" }}
  type="text"
  name="kolom1"
  placeholder="Isi Kolom1"
  onChange={(e) => setKolom1(e.target.value)}
/>
<br></br>
<br></br>
<label htmlFor="kolom2">Kolom2 :</label>
<input
  style={{ marginLeft: "1em" }}
  type="text"
  name="kolom2"
  placeholder="Isi Kolom2"
  onChange={(e) => setKolom2(e.target.value)}
/>
<br></br>
<br></br>
<button onClick={onSimpanDokDgId}>SIMPAN DOK DG ID</button>
</fieldset>
<br></br>
<fieldset>
  <legend>Set Data dg merge: true </legend>
  <label htmlFor="kolom3">Kolom3 :</label>
  <input
    style={{ marginLeft: "1em" }}
    type="text"
    name="kolom3"
    placeholder="Isi Kolom3"
    onChange={(e) => setKolom3(e.target.value)}
  />
  <br></br>
  <br></br>
  <button onClick={onSimpanDokDgIdMerge}>SIMPAN DOK DG ID MERGE<
/fieldset>

```



```

    </fieldset>
  </div>
);
}

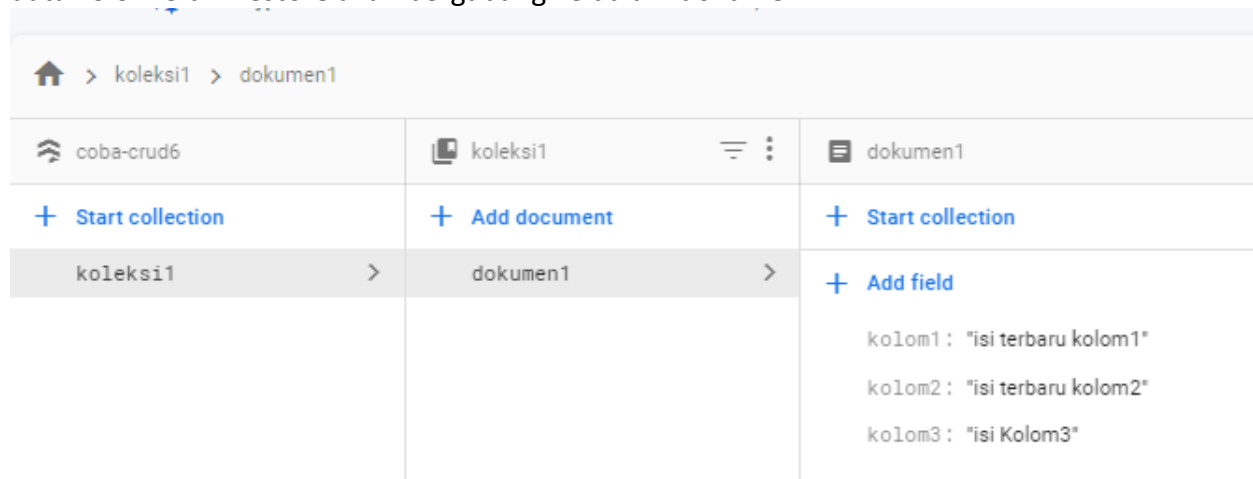
export default Latihan1;

```

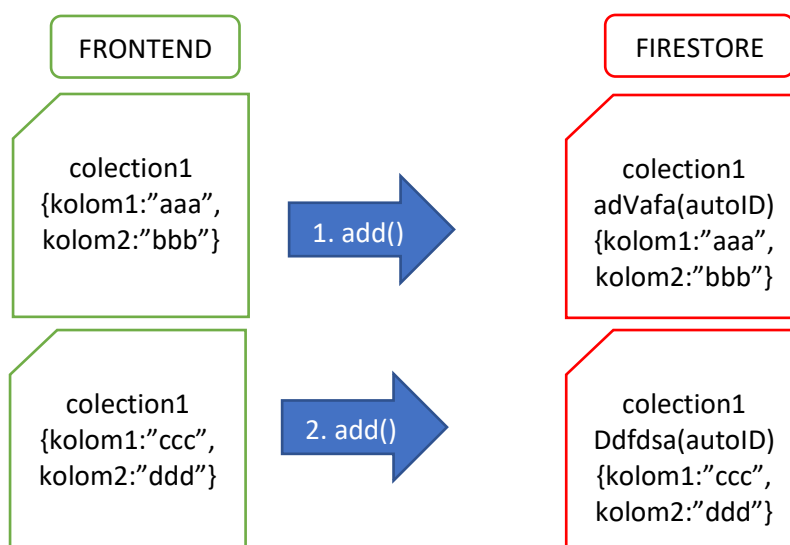
Data yang dikirim

Nama Collection	Koleksi1
Nama Dokumen	Dokumen1
Kolom3	Isi kolom3

data kolom 3 di firestore akan bergabung ke dalam dokumen1



2.3c. Write Collection dengan ID Dokumen otomatis => add



Tetapi, terkadang tidak ada ID yang berarti untuk dokumen ini dan lebih mudah untuk membiarkan Cloud Firestore membuat ID secara otomatis untuk Anda. Anda dapat melakukannya dengan memanggil `add()`:

```
const onAddCollection = () => {  
  // if add again and again => generate document with auto ID  
  console.log("BuatCollection", addNameCollection, addCollection);  
  db.collection(addNameCollection)  
    .add({  
      ...addCollection,  
    })  
    .then((doc) => {  
      console.log("Added doc with ID : ", doc.id);  
    });  
};
```

<https://github.com/edylee/fire-crud/blob/latihan2/admin-crud/src/Latihan2.js>

Latihan 2 Menambah Data dengan ID Dokumen Otomatis

Set Data :

Nama Collection :

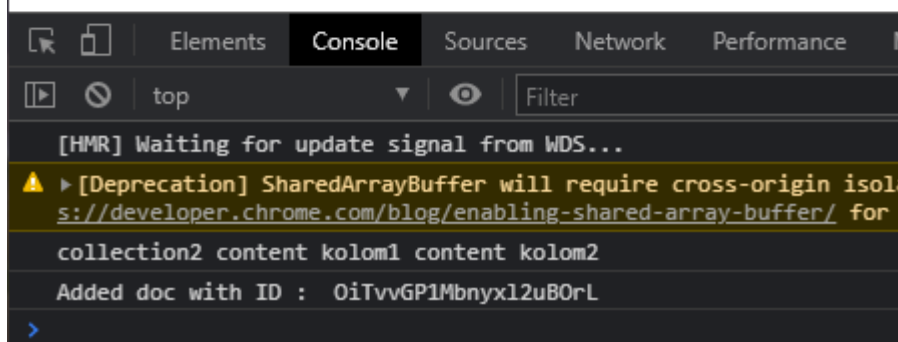
Nama Dokumen :

Kolom1 :

Kolom2 :

Set Data dg merge: true

Kolom3 :



```
//src/Latihan2.js
import React, { useState } from "react";
//import firebase
import { db } from "../firebase";

function Latihan2() {
  //state
  const [nmCollection, setNmCollection] = useState("");
  const [nmDokumen, setNmDokumen] = useState("");
  const [kolom1, setKolom1] = useState("");
  const [kolom2, setKolom2] = useState("");
  const [kolom3, setKolom3] = useState("");
  //state loading pada proses asynchronous
  const [loading, setLoading] = useState(false);
  //state untuk data dari hasil pembacaan

  //button simpan function
  function onSimpanDok() {
```

```

    if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
    if (kolom1 === "") return console.log("Kolom1 Kosong");
    if (kolom2 === "") return console.log("Kolom2 Kosong");
    console.log(nmCollection, kolom1, kolom2);
    setLoading(true);
    db.collection(nmCollection)
        .add({
            kolom1: kolom1,
            kolom2: kolom2,
        })
        .then((doc) => {
            console.log("Added doc with ID : ", doc.id);
            setLoading(false);
        })
        .catch((error) => {
            console.error("Error writing document: ", error);
        });
}

function onSimpanDokDgIdMerge() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
    if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
    if (kolom3 === "") return console.log("Kolom3 Kosong");
    console.log(nmCollection, nmDokumen, kolom3);
    setLoading(true);
    db.collection(nmCollection)
        .doc(nmDokumen)
        .set(
            {
                kolom3: kolom3,
            },
            { merge: true }
        )
        .then(() => {
            console.log("Document successfully written!");
            setLoading(false);
        })
        .catch((error) => {
            console.error("Error writing document: ", error);
        });
}

if (loading) {
    return <>Tunggu Masih Proses, Loading...</>;
}

return (

```

```

<div>
  <h3>Latihan 2 Menambah Data dengan ID Dokumen Otomatis</h3>
  <fieldset>
    <legend>Set Data :</legend>
    <label htmlFor="nmCollection">Nama Collection :</label>
    <input
      style={{ marginLeft: "1em" }}
      type="text"
      name="nmCollection"
      placeholder="Isi Nama Collection"
      onChange={(e) => setNmCollection(e.target.value)}
    />
    <br></br>
    <br></br>
    <label htmlFor="nmDokumen">Nama Dokumen :</label>
    <input
      style={{ marginLeft: "1em" }}
      type="text"
      name="nmDokumen"
      placeholder="Isi Kolom1"
      onChange={(e) => setNmDokumen(e.target.value)}
    />
    <br></br>
    <br></br>
    <label htmlFor="kolom1">Kolom1 :</label>
    <input
      style={{ marginLeft: "1em" }}
      type="text"
      name="kolom1"
      placeholder="Isi Kolom1"
      onChange={(e) => setKolom1(e.target.value)}
    />
    <br></br>
    <br></br>
    <label htmlFor="kolom2">Kolom2 :</label>
    <input
      style={{ marginLeft: "1em" }}
      type="text"
      name="kolom2"
      placeholder="Isi Kolom2"
      onChange={(e) => setKolom2(e.target.value)}
    />
    <br></br>
    <br></br>
    <button onClick={onSimpanDok}>SIMPAN DOK</button>
  </fieldset>
  <br></br>
  <fieldset>
    <legend>Set Data dg merge: true </legend>

```

```

        <label htmlFor="kolom3">Kolom3 :</label>
        <input
          style={{ marginLeft: "1em" }}
          type="text"
          name="kolom3"
          placeholder="Isi Kolom3"
          onChange={(e) => setKolom3(e.target.value)}
        />
        <br></br>
        <br></br>
        <button onClick={onSimpanDokDgIdMerge}>SIMPAN DOK DG ID MERGE<
    /button>
    </fieldset>
  </div>
);
}

export default Latihan2;

```

perubahan file App.js

```

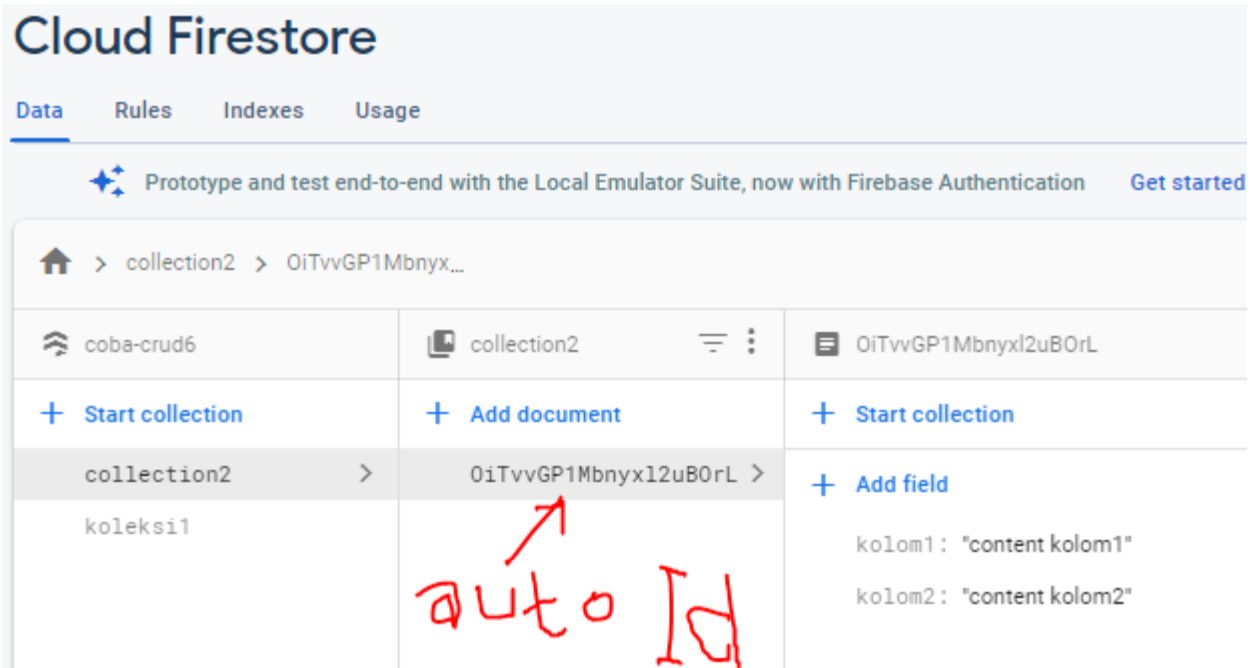
//src/App.js
// import Latihan1 from "../Latihan1";
import Latihan2 from "../Latihan2";
function App() {
  return (
    <div>
      {/* <Latihan1 /> */}
      <Latihan2 />
    </div>
  );
}

export default App;

```

Data yang dikirim

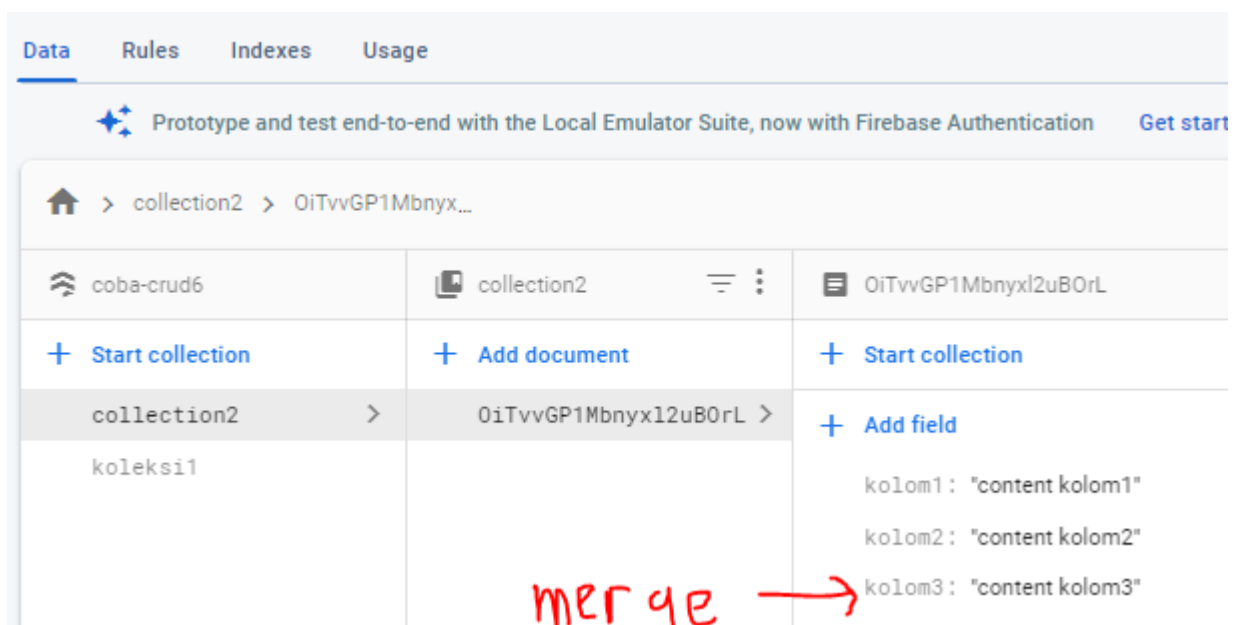
Nama Collection	Collection2
Kolom1	Content kolom1
Kolom2	Content kolom2



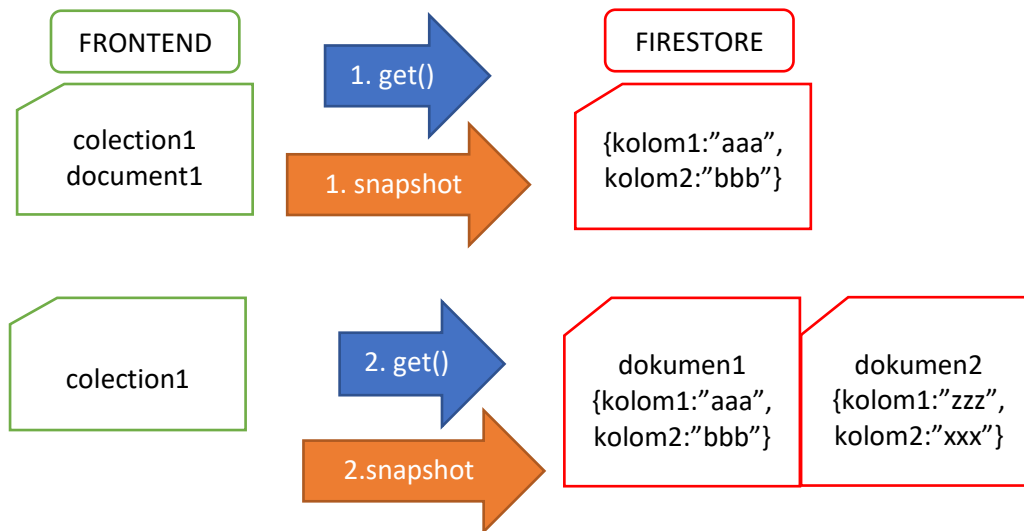
dokumen secara langsung diberikan ID (OiTvGP1Mbnyx12uBOrL) oleh firestore.
Perintah set merge juga bisa digunakan untuk menambahkan field dalam firestore.

Data yang dikirim => "SIMPAN DOK DG ID MERGE"

Nama Collection Collection2
Nama Dokumen OiTvGP1Mbnyx12uBOrL
Kolom3 Content kolom3



2. READ DATA COLLECTION KE FIRESTORE (DATA SEKALI)



Untuk melihat data seluruh collection maka harus dikirimkan nama collection dalam perintah ke firebase, hasil dari data tersebut berupa sebuah array object, yang kemudian dilakukan mapping atau foreach untuk mendapatkan datanya.

<https://firebase.google.com/docs/firestore/query-data/get-data?authuser=0>

Mendapatkan dokumen

Contoh berikut menunjukkan cara mengambil konten dari sebuah dokumen menggunakan `get()`:

```
var docRef = db.collection("cities").doc("SF");

docRef
  .get()
  .then((doc) => {
    if (doc.exists) {
      console.log("Document data:", doc.data());
    } else {
      // doc.data() will be undefined in this case
      console.log("No such document!");
    }
  })
  .catch((error) => {
    console.log("Error getting document:", error);
  });
```

Pada Latihan1 kita masukkan perintah tersebut kedalam tombol LIHAT dan kita tampilkan dalam JSON.

<https://github.com/edycoleee/fire-crud/blob/latihan3/admin-crud/src/Latihan3.js>


```

//src/Latihan3.js
import React, { useState } from "react";
//import firebase
import { db } from "../firebase";

function Latihan3() {
  //state
  const [nmCollection, setNmCollection] = useState("");
  const [nmDokumen, setNmDokumen] = useState("");
  const [kolom1, setKolom1] = useState("");
  const [kolom2, setKolom2] = useState("");
  const [kolom3, setKolom3] = useState("");

  //state untuk data dari hasil pembacaan
  const [dataCollection, setDataCollection] = useState([]);

  //button simpan fuction
  function onSimpanDokDgId() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
    if (kolom1 === "") return console.log("Kolom1 Kosong");
    if (kolom2 === "") return console.log("Kolom2 Kosong");
    console.log(nmCollection, nmDokumen, kolom1, kolom2);
    db.collection(nmCollection)
      .doc(nmDokumen)
      .set({
        kolom1: kolom1,
        kolom2: kolom2,
      })
      .then(() => {
        console.log("Document successfully written!");
      })
      .catch((error) => {
        console.error("Error writing document: ", error);
      });
  }

  function onSimpanDokDgIdMerge() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
    if (kolom3 === "") return console.log("Kolom3 Kosong");
    console.log(nmCollection, nmDokumen, kolom1, kolom2);
    db.collection(nmCollection)
      .doc(nmDokumen)
      .set(
        {
          kolom3: kolom3,
        }
      );
  }
}

```

```

    },
    { merge: true }
  )
  .then(() => {
    console.log("Document successfully written!");
  })
  .catch((error) => {
    console.error("Error writing document: ", error);
  });
}

function onSimpanDok() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
;
  if (kolom1 === "") return console.log("Kolom1 Kosong");
  if (kolom2 === "") return console.log("Kolom2 Kosong");
  console.log(nmCollection, kolom1, kolom2);
  db.collection(nmCollection)
    .add({
      kolom1: kolom1,
      kolom2: kolom2,
    })
    .then((doc) => {
      console.log("Added doc with ID : ", doc.id);
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}

const getAllCol = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
;
  console.log("Get All Data Collection", nmCollection);
  db.collection(nmCollection)
    .get()
    .then((firecol) => {
      const data = firecol.docs.map((doc) => ({
        id: doc.id,
        ...doc.data(),
      }));
      console.log("Get All Data Collection :", data);
      setDataCollection(data);
    })
    .catch((error) => console.error("Error Get Data :", error));
};

const getDok = () => {

```

```

    if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
    if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
    console.log("Get Dokumen", nmDokumen );
};

return (
  <div>
    <h3>Latihan 3 Melihat Data Sekali</h3>
    <fieldset>
      <legend>Set Data :</legend>
      <label htmlFor="nmCollection">Nama Collection :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="nmCollection"
        placeholder="Isi Nama Collection"
        onChange={(e) => setNmCollection(e.target.value)}
      />{ " "}
      <button onClick={getAllCol}>LIHAT COLL</button>
      <br></br>
      <br></br>
      <label htmlFor="nmDokumen">Nama Dokumen :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="nmDokumen"
        placeholder="Isi Kolom1"
        onChange={(e) => setNmDokumen(e.target.value)}
      />{ " "}
      <button onClick={getDok}>LIHAT DOK</button>
      <br></br>
      <br></br>
      <label htmlFor="kolom1">Kolom1 :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="kolom1"
        placeholder="Isi Kolom1"
        onChange={(e) => setKolom1(e.target.value)}
      />
      <br></br>
      <br></br>
      <label htmlFor="kolom2">Kolom2 :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="kolom2"
        placeholder="Isi Kolom2"

```

```

        onChange={(e) => setKolom2(e.target.value)}
      />
      <br></br>
      <br></br>
      <button onClick={onSimpanDokDgId}>SIMPAN DOK DG ID</button>{"
    "}
    <button onClick={onSimpanDok}>SIMPAN DOK</button>
  </fieldset>
  <br></br>
  <fieldset>
    <legend>Set Data dg merge: true </legend>
    <label htmlFor="kolom3">Kolom3 :</label>
    <input
      style={{ marginLeft: "1em" }}
      type="text"
      name="kolom3"
      placeholder="Isi Kolom3"
      onChange={(e) => setKolom3(e.target.value)}
    />
    <br></br>
    <br></br>
    <button onClick={onSimpanDokDgIdMerge}>SIMPAN DOK DG ID MERGE<
  /button>
</fieldset>
<br></br>
<fieldset>
  <legend>Data Firestore </legend>
  {JSON.stringify(dataCollection)}
</fieldset>
</div>
);
}

export default Latihan3;

```

Perubahan pada file App.js

```

//src/App.js
// import Latihan1 from "./Latihan1";
//import Latihan2 from "./Latihan2";
import Latihan3 from "./Latihan3";
function App() {
  return (
    <div>
      {/* <Latihan1 /> */}
      {/* <Latihan2 /> */}
      <Latihan3 />
    </div>
  );
}

```

```
}
```

```
export default App;
```

Latihan 3 Melihat Data Sekali

Set Data :

Nama Collection :

Nama Dokumen :

Kolom1 :

Kolom2 :

Set Data dg merge: true

Kolom3 :

Data Firestore

[{"id":"dokumen1","kolom3":"ddddd","kolom1":"aaa","kolom2":"sss"}, {"id":"dokumen2","kolom1":"aaa","kolom2":"ssss"}]

Elements

Console

Sources

Network

Performance

Memory

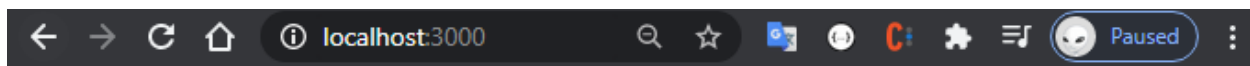
>>

3. READ DATA COLLECTION DARI FIRESTORE (REALTIME)

<https://firebase.google.com/docs/firestore/query-data/listen?authuser=0>

```
db.collection("cities")
  .doc("SF")
  .onSnapshot((doc) => {
    console.log("Current data: ", doc.data());
  });
```

pada contoh diatas snapshot biasanya digunakan dalam useEffect yang akan secara otomatis update data jika ada perubahan isi database, tanpa harus mengirimkan perintah lihat ulang kedatabase. (KELEBIHAN FIRESTORE)



Latihan 3 Melihat Data Sekali

Set Data :

Nama Collection :

Nama Dokumen :

Kolom1 :

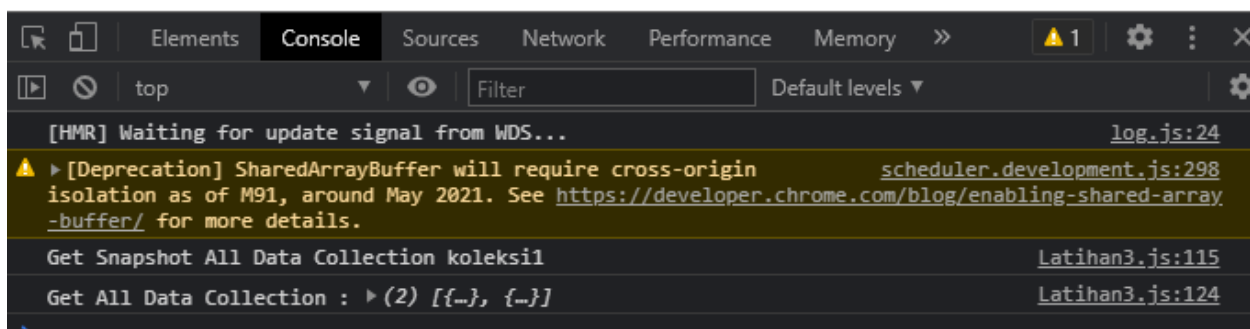
Kolom2 :

Set Data dg merge: true

Kolom3 :

Data Firestore

[{"id":"dokumen1","kolom1":"aaa","kolom3":"ddddd","kolom2":"sss"}, {"id":"dokumen2","kolom1":"fff","kolom2":"ssss"}]



```
//src/Latihan3.js
import React, { useState } from "react";
//import firebase
import { db } from "../firebase";

function Latihan3() {
  //state
  const [nmCollection, setNmCollection] = useState("");
  const [nmDokumen, setNmDokumen] = useState("");
  const [kolom1, setKolom1] = useState("");
  const [kolom2, setKolom2] = useState("");
  const [kolom3, setKolom3] = useState("");

  //state untuk data dari hasil pembacaan
  const [dataCollection, setDataCollection] = useState([]);
```

```

//button simpan fuction
function onSimpanDokDgId() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
  if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
  if (kolom1 === "") return console.log("Kolom1 Kosong");
  if (kolom2 === "") return console.log("Kolom2 Kosong");
  console.log(nmCollection, nmDokumen, kolom1, kolom2);
  db.collection(nmCollection)
    .doc(nmDokumen)
    .set({
      kolom1: kolom1,
      kolom2: kolom2,
    })
    .then(() => {
      console.log("Document successfully written!");
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}

function onSimpanDokDgIdMerge() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
  if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
  if (kolom3 === "") return console.log("Kolom3 Kosong");
  console.log(nmCollection, nmDokumen, kolom1, kolom2);
  db.collection(nmCollection)
    .doc(nmDokumen)
    .set(
      {
        kolom3: kolom3,
      },
      { merge: true }
    )
    .then(() => {
      console.log("Document successfully written!");
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}

function onSimpanDok() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
  if (kolom1 === "") return console.log("Kolom1 Kosong");
  if (kolom2 === "") return console.log("Kolom2 Kosong");

```

```

    console.log(nmCollection, kolom1, kolom2);
    db.collection(nmCollection)
      .add({
        kolom1: kolom1,
        kolom2: kolom2,
      })
      .then((doc) => {
        console.log("Added doc with ID : ", doc.id);
      })
      .catch((error) => {
        console.error("Error writing document: ", error);
      });
  }

  const getAllCol = () => {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
  };

  console.log("Get Once All Data Collection", nmCollection);
  db.collection(nmCollection)
    .get()
    .then((firecol) => {
      const data = firecol.docs.map((doc) => ({
        id: doc.id,
        ...doc.data(),
      }));
      console.log("Get All Data Collection :", data);
      setDataCollection(data);
    })
    .catch((error) => console.error("Error Get Data :", error));
};

const getAllColSnapshot = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
};

console.log("Get Snapshot All Data Collection", nmCollection);
db.collection(nmCollection).onSnapshot((firecol) => {
  const data = firecol.docs.map((doc) => ({
    id: doc.id,
    ...doc.data(),
  }));
  console.log("Get All Data Collection :", data);
  setDataCollection(data);
});

const getAllColSnapshotPush = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
};

console.log("Get Snapshot All Data Collection", nmCollection);

```



```

db.collection(nmCollection).onSnapshot((firecol) => {
  let data = [];
  firecol.forEach((doc) => {
    data.push({
      id: doc.id,
      ...doc.data(),
    });
  });
  console.log("Get All Data Collection :", data);
  setDataCollection(data);
});
};
const getDok = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
  console.log("Get Dokumen", nmCollection);
};

return (
  <div>
    <h3>Latihan 3 Melihat Data Sekali</h3>
    <fieldset>
      <legend>Set Data </legend>
      <label htmlFor="nmCollection">Nama Collection </label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="nmCollection"
        placeholder="Isi Nama Collection"
        onChange={(e) => setNmCollection(e.target.value)}
      />{" "}
      <button onClick={getAllCol}>LIHAT COLL</button>{" "}
      <button onClick={getAllColSnapshot}>SNAPSHOT COLL1</button>{"
    "}
      <button onClick={getAllColSnapshotPush}>SNAPSHOT COLL2</button>
    </div>

    <br></br>
    <br></br>
    <label htmlFor="nmDokumen">Nama Dokumen </label>
    <input
      style={{ marginLeft: "1em" }}
      type="text"
      name="nmDokumen"
      placeholder="Isi Kolom1"
      onChange={(e) => setNmDokumen(e.target.value)}
    />{" "}
    <button onClick={getDok}>LIHAT DOK</button>
    <br></br>
    <br></br>
  )
)

```

```

<label htmlFor="kolom1">Kolom1 :</label>
<input
  style={{ marginLeft: "1em" }}
  type="text"
  name="kolom1"
  placeholder="Isi Kolom1"
  onChange={(e) => setKolom1(e.target.value)}
/>
<br></br>
<br></br>
<label htmlFor="kolom2">Kolom2 :</label>
<input
  style={{ marginLeft: "1em" }}
  type="text"
  name="kolom2"
  placeholder="Isi Kolom2"
  onChange={(e) => setKolom2(e.target.value)}
/>
<br></br>
<br></br>
<button onClick={onSimpanDokDgId}>SIMPAN DOK DG ID</button>{"
"}
  <button onClick={onSimpanDok}>SIMPAN DOK</button>
</fieldset>
<br></br>
<fieldset>
  <legend>Set Data dg merge: true </legend>
  <label htmlFor="kolom3">Kolom3 :</label>
  <input
    style={{ marginLeft: "1em" }}
    type="text"
    name="kolom3"
    placeholder="Isi Kolom3"
    onChange={(e) => setKolom3(e.target.value)}
  />
  <br></br>
  <br></br>
  <button onClick={onSimpanDokDgIdMerge}>SIMPAN DOK DG ID MERGE<
/button>
</fieldset>
<br></br>
<fieldset>
  <legend>Data Firestore </legend>
  {JSON.stringify(dataCollection)}
</fieldset>
</div>
);
}

```

```
export default Latihan3;
```

4. READ DATA DOKUMEN DARI FIRESTORE (REALTIME)

Untuk melihat data dokumen maka nama collection dan nama dokumen harus dikirimkan dalam perintah ke firebase, hasil dari data tersebut berupa sebuah object

Latihan 3 Melihat Data Sekali

Set Data :

Nama Collection :

Nama Dokumen :

Kolom1 :

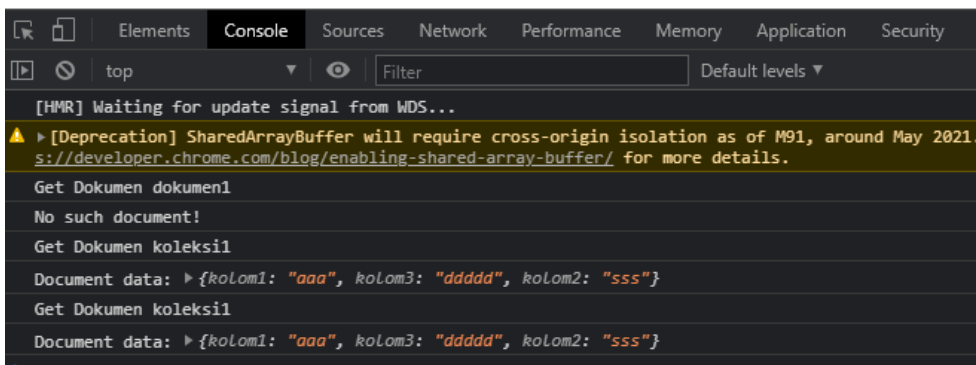
Kolom2 :

Set Data dg merge: true

Kolom3 :

Data Firestore

```
{"kolom1": "aaa", "kolom3": "dddd", "kolom2": "sss"}
```



<https://github.com/edylee/fire-crud/blob/latihan3a/admin-crud/src/Latihan3.js>

```
//src/Latihan3.js
import React, { useState } from "react";
//import firebase
import { db } from "../firebase";

function Latihan3() {
  //state
  const [nmCollection, setNmCollection] = useState("");
  const [nmDokumen, setNmDokumen] = useState("");
  const [kolom1, setKolom1] = useState("");
```

```

const [kolom2, setKolom2] = useState("");
const [kolom3, setKolom3] = useState("");

//state untuk data dari hasil pembacaan
const [dataCollection, setDataCollection] = useState([]);

//button simpan fuction
function onSimpanDokDgId() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
;
  if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
  if (kolom1 === "") return console.log("Kolom1 Kosong");
  if (kolom2 === "") return console.log("Kolom2 Kosong");
  console.log(nmCollection, nmDokumen, kolom1, kolom2);
  db.collection(nmCollection)
    .doc(nmDokumen)
    .set({
      kolom1: kolom1,
      kolom2: kolom2,
    })
    .then(() => {
      console.log("Document successfully written!");
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}

function onSimpanDokDgIdMerge() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
;
  if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
  if (kolom3 === "") return console.log("Kolom3 Kosong");
  console.log(nmCollection, nmDokumen, kolom1, kolom2);
  db.collection(nmCollection)
    .doc(nmDokumen)
    .set(
      {
        kolom3: kolom3,
      },
      { merge: true }
    )
    .then(() => {
      console.log("Document successfully written!");
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}

```

```

function onSimpanDok() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
  if (kolom1 === "") return console.log("Kolom1 Kosong");
  if (kolom2 === "") return console.log("Kolom2 Kosong");
  console.log(nmCollection, kolom1, kolom2);
  db.collection(nmCollection)
    .add({
      kolom1: kolom1,
      kolom2: kolom2,
    })
    .then((doc) => {
      console.log("Added doc with ID : ", doc.id);
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}

const getAllCol = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
  console.log("Get Once All Data Collection", nmCollection);
  db.collection(nmCollection)
    .get()
    .then((firecol) => {
      const data = firecol.docs.map((doc) => ({
        id: doc.id,
        ...doc.data(),
      }));
      console.log("Get All Data Collection :", data);
      setDataCollection(data);
    })
    .catch((error) => console.error("Error Get Data :", error));
};

const getAllColSnapshot = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
  console.log("Get Snapshot All Data Collection", nmCollection);
  db.collection(nmCollection).onSnapshot((firecol) => {
    const data = firecol.docs.map((doc) => ({
      id: doc.id,
      ...doc.data(),
    }));
    console.log("Get All Data Collection :", data);
    setDataCollection(data);
  });
};

```

```

};

const getAllColSnapshotPush = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
  console.log("Get Snapshot All Data Collection", nmCollection);
  db.collection(nmCollection).onSnapshot((firecol) => {
    let data = [];
    firecol.forEach((doc) => {
      data.push({
        id: doc.id,
        ...doc.data(),
      });
    });
    console.log("Get All Data Collection :", data);
    setDataCollection(data);
  });
};

const getDok = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
  if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
  console.log("Get Dokumen", nmCollection);
  db.collection(nmCollection)
    .doc(nmDokumen)
    .get()
    .then((doc) => {
      if (doc.exists) {
        console.log("Document data:", doc.data());
        setDataCollection(doc.data());
      } else {
        // doc.data() will be undefined in this case
        console.log("No such document!");
      }
    })
    .catch((error) => {
      console.log("Error getting document:", error);
    });
};

const getDokSnapshot = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
  if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
  console.log("Get Dokumen", nmCollection);
  db.collection(nmCollection)
    .doc(nmDokumen)
    .onSnapshot((doc) => {
      if (doc.exists) {

```

```

        console.log("Document data:", doc.data());
        setDataCollection(doc.data());
    } else {
        // doc.data() will be undefined in this case
        console.log("No such document!");
    }
});

};

return (
    <div>
        <h3>Latihan 3 Melihat Data Sekali</h3>
        <fieldset>
            <legend>Set Data :</legend>
            <label htmlFor="nmCollection">Nama Collection :</label>
            <input
                style={{ marginLeft: "1em" }}
                type="text"
                name="nmCollection"
                placeholder="Isi Nama Collection"
                onChange={(e) => setNmCollection(e.target.value)}
            />{" "}
            <button onClick={getAllCol}>LIHAT COLL</button>{" "}
            <button onClick={getAllColSnapshot}>SNAPSHOT COLL1</button>{"
"
            <button onClick={getAllColSnapshotPush}>SNAPSHOT COLL2</button>
"
        <br></br>
        <br></br>
        <label htmlFor="nmDokumen">Nama Dokumen :</label>
        <input
            style={{ marginLeft: "1em" }}
            type="text"
            name="nmDokumen"
            placeholder="Isi Kolom1"
            onChange={(e) => setNmDokumen(e.target.value)}
        />{" "}
        <button onClick={getDok}>LIHAT DOK</button>{" "}
        <button onClick={getDokSnapshot}>SNAPSHOT DOK</button> <br></b
r>
        <br></br>
        <label htmlFor="kolom1">Kolom1 :</label>
        <input
            style={{ marginLeft: "1em" }}
            type="text"
            name="kolom1"
            placeholder="Isi Kolom1"
            onChange={(e) => setKolom1(e.target.value)}
        />
    </div>
);

```

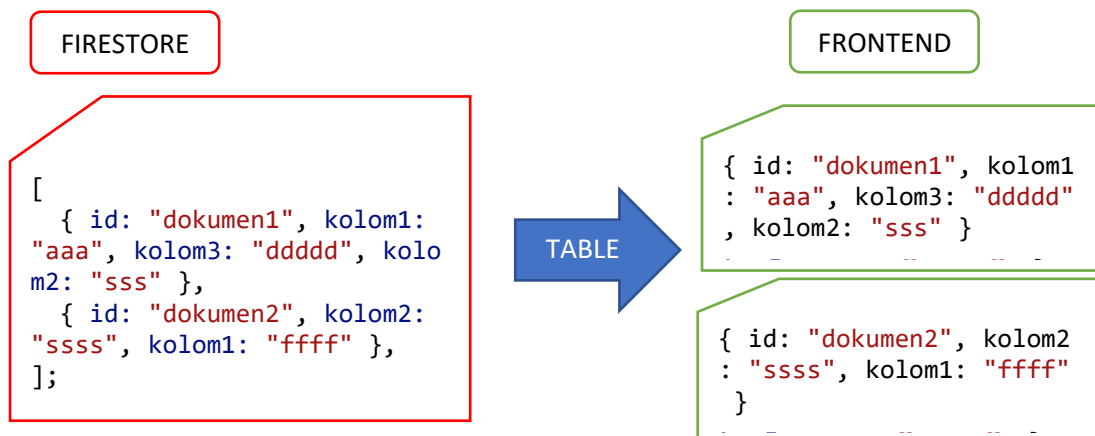
```

        <br></br>
        <br></br>
        <label htmlFor="kolom2">Kolom2 :</label>
        <input
            style={{ marginLeft: "1em" }}
            type="text"
            name="kolom2"
            placeholder="Isi Kolom2"
            onChange={(e) => setKolom2(e.target.value)}
        />
        <br></br>
        <br></br>
        <button onClick={onSimpanDokDgId}>SIMPAN DOK DG ID</button>{"
    "
        <button onClick={onSimpanDok}>SIMPAN DOK</button>
    </fieldset>
    <br></br>
    <fieldset>
        <legend>Set Data dg merge: true </legend>
        <label htmlFor="kolom3">Kolom3 :</label>
        <input
            style={{ marginLeft: "1em" }}
            type="text"
            name="kolom3"
            placeholder="Isi Kolom3"
            onChange={(e) => setKolom3(e.target.value)}
        />
        <br></br>
        <br></br>
        <button onClick={onSimpanDokDgIdMerge}>SIMPAN DOK DG ID MERGE<
    /button>
    </fieldset>
    <br></br>
    <fieldset>
        <legend>Data Firestore </legend>
        {JSON.stringify(dataCollection)}
    </fieldset>
    </div>
    );
}

export default Latihan3;

```


5. MENAMPILKAN DATA KEDALAM TABEL



Mapping data Array object menjadi single object untuk ditampilkan kedalam satu baris data di table

Perubahan dalam file App.js

```
//src/App.js
// import Latihan1 from "./Latihan1";
//import Latihan2 from "./Latihan2";
// import Latihan3 from "./Latihan3";
import Latihan4 from "./Latihan4";
function App() {
  return (
    <div>
      {/* <Latihan1 /> */}
      {/* <Latihan2 /> */}
      {/* <Latihan3 /> */}
      <Latihan4 />
    </div>
  );
}
export default App;
```

Penyempurnaan bentuk dari get data dokumen agar berbentuk sama seperti get collection, sehingga dapat ditampilkan dalam table yang sama bentuknya.

<https://github.com/edycoleee/fire-crud/blob/latihan4/admin-crud/src/Latihan4.js>

Latihan 4 Menampilkan Tabel

Set Data :

Nama Collection :

Nama Dokumen :

Kolom1 :

Kolom2 :

Set Data dg merge: true

Kolom3 :

Data Firestore

[{"id":"dokumen1","kolom2":"sss","kolom3":"ddddd","kolom1":"aaa"},
{"id":"dokumen2","kolom1":"ffff","kolom2":"ssss"}]

Table Firestore

kolom 1	kolom 2	kolom 3
aaa	sss	ddddd
ffff	ssss	

```
//src/Latihan4.js
import React, { useState } from "react";
//import firebase
import { db } from "../firebase";

function Latihan4() {
  //state
  const [nmCollection, setNmCollection] = useState("");
  const [nmDokumen, setNmDokumen] = useState("");
  const [kolom1, setKolom1] = useState("");
  const [kolom2, setKolom2] = useState("");
  const [kolom3, setKolom3] = useState("");

  //state untuk data dari hasil pembacaan
  const [dataCollection, setDataCollection] = useState([]);

  //button simpan fuction
  function onSimpanDokDgId() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
    if (kolom1 === "") return console.log("Kolom1 Kosong");
    if (kolom2 === "") return console.log("Kolom2 Kosong");
    console.log(nmCollection, nmDokumen, kolom1, kolom2);
    db.collection(nmCollection)
      .doc(nmDokumen)
      .set({
```

```

        kolom1: kolom1,
        kolom2: kolom2,
    })
    .then(() => {
        console.log("Document successfully written!");
    })
    .catch((error) => {
        console.error("Error writing document: ", error);
    });
}

function onSimpanDokDgIdMerge() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
;
    if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
    if (kolom3 === "") return console.log("Kolom3 Kosong");
    console.log(nmCollection, nmDokumen, kolom1, kolom2);
    db.collection(nmCollection)
        .doc(nmDokumen)
        .set(
            {
                kolom3: kolom3,
            },
            { merge: true }
        )
        .then(() => {
            console.log("Document successfully written!");
        })
        .catch((error) => {
            console.error("Error writing document: ", error);
        });
}

function onSimpanDok() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
;
    if (kolom1 === "") return console.log("Kolom1 Kosong");
    if (kolom2 === "") return console.log("Kolom2 Kosong");
    console.log(nmCollection, kolom1, kolom2);
    db.collection(nmCollection)
        .add({
            kolom1: kolom1,
            kolom2: kolom2,
        })
        .then((doc) => {
            console.log("Added doc with ID : ", doc.id);
        })
        .catch((error) => {
            console.error("Error writing document: ", error);
        });
}

```

```

    });
  }

  const getAllCol = () => {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
    console.log("Get Once All Data Collection", nmCollection);
    db.collection(nmCollection)
      .get()
      .then((firecol) => {
        const data = firecol.docs.map((doc) => ({
          id: doc.id,
          ...doc.data(),
        }));
        console.log("Get All Data Collection :", data);
        setDataCollection(data);
      })
      .catch((error) => console.error("Error Get Data :", error));
  };

  const getAllColSnapshot = () => {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
    console.log("Get Snapshot All Data Collection", nmCollection);
    db.collection(nmCollection).onSnapshot((firecol) => {
      const data = firecol.docs.map((doc) => ({
        id: doc.id,
        ...doc.data(),
      }));
      console.log("Get All Data Collection :", data);
      setDataCollection(data);
    });
  };

  const getAllColSnapshotPush = () => {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
    console.log("Get Snapshot All Data Collection", nmCollection);
    db.collection(nmCollection).onSnapshot((firecol) => {
      let data = [];
      firecol.forEach((doc) => {
        data.push({
          id: doc.id,
          ...doc.data(),
        });
      });
      console.log("Get All Data Collection :", data);
      setDataCollection(data);
    });
  };

```

```

};
const getDok = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
  if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
  console.log("Get Dokumen", nmCollection);
  db.collection(nmCollection)
    .doc(nmDokumen)
    .get()
    .then((doc) => {
      if (!doc.exists) return console.log("No such document!");
      return setDataCollection([{ id: doc.id, ...doc.data() }]);
    })
    .catch((error) => {
      console.log("Error getting document:", error);
    });
};

const getDokSnapshot = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong")
;
  if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
  console.log("Get Dokumen", nmCollection);
  db.collection(nmCollection)
    .doc(nmDokumen)
    .onSnapshot((doc) => {
      if (!doc.exists) return console.log("No such document!");
      return setDataCollection([{ id: doc.id, ...doc.data() }]);
    });
};

return (
  <div>
    <h3>Latihan 4 Menampilkan Tabel</h3>
    <fieldset>
      <legend>Set Data </legend>
      <label htmlFor="nmCollection">Nama Collection </label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="nmCollection"
        placeholder="Isi Nama Collection"
        onChange={(e) => setNmCollection(e.target.value)}
      />{" "}
      <button onClick={getAllCol}>LIHAT COLL</button>{" "}
      <button onClick={getAllColSnapshot}>SNAPSHOT COLL1</button>{"
    <button onClick={getAllColSnapshotPush}>SNAPSHOT COLL2</button
  </div>
)

```

```

<br></br>
<br></br>
<label htmlFor="nmDokumen">Nama Dokumen :</label>
<input
  style={{ marginLeft: "1em" }}
  type="text"
  name="nmDokumen"
  placeholder="Isi Kolom1"
  onChange={(e) => setNmDokumen(e.target.value)}
/>{" "}
<button onClick={getDok}>LIHAT DOK</button>{" "}
<button onClick={getDokSnapshot}>SNAPSHOT DOK</button> <br></b
r>

<br></br>
<label htmlFor="kolom1">Kolom1 :</label>
<input
  style={{ marginLeft: "1em" }}
  type="text"
  name="kolom1"
  placeholder="Isi Kolom1"
  onChange={(e) => setKolom1(e.target.value)}
/>
<br></br>
<br></br>
<label htmlFor="kolom2">Kolom2 :</label>
<input
  style={{ marginLeft: "1em" }}
  type="text"
  name="kolom2"
  placeholder="Isi Kolom2"
  onChange={(e) => setKolom2(e.target.value)}
/>
<br></br>
<br></br>
<button onClick={onSimpanDokDgId}>SIMPAN DOK DG ID</button>{"
"}

  <button onClick={onSimpanDok}>SIMPAN DOK</button>
</fieldset>
<br></br>
<fieldset>
  <legend>Set Data dg merge: true </legend>
  <label htmlFor="kolom3">Kolom3 :</label>
  <input
    style={{ marginLeft: "1em" }}
    type="text"
    name="kolom3"
    placeholder="Isi Kolom3"
    onChange={(e) => setKolom3(e.target.value)}
  />

```

```

        <br></br>
        <br></br>
        <button onClick={onSimpanDokDgIdMerge}>SIMPAN DOK DG ID MERGE<
/button>
    </fieldset>
    <br></br>
    <fieldset>
        <legend>Data Firestore </legend>
        {JSON.stringify(dataCollection)}
    </fieldset>
    <fieldset>
        <legend>Table Firestore </legend>
        <table>
            <thead>
                <tr>
                    <th>kolom 1</th>
                    <th>kolom 2</th>
                    <th>kolom 3</th>
                </tr>
            </thead>
            <tbody>
                {dataCollection?.map((row) => (
                    <tr key={row.id}>
                        <td>{row.kolom1}</td>
                        <td>{row.kolom2}</td>
                        <td>{row.kolom3}</td>
                    </tr>
                ))}
            </tbody>
        </table>
    </fieldset>
</div>
);
}

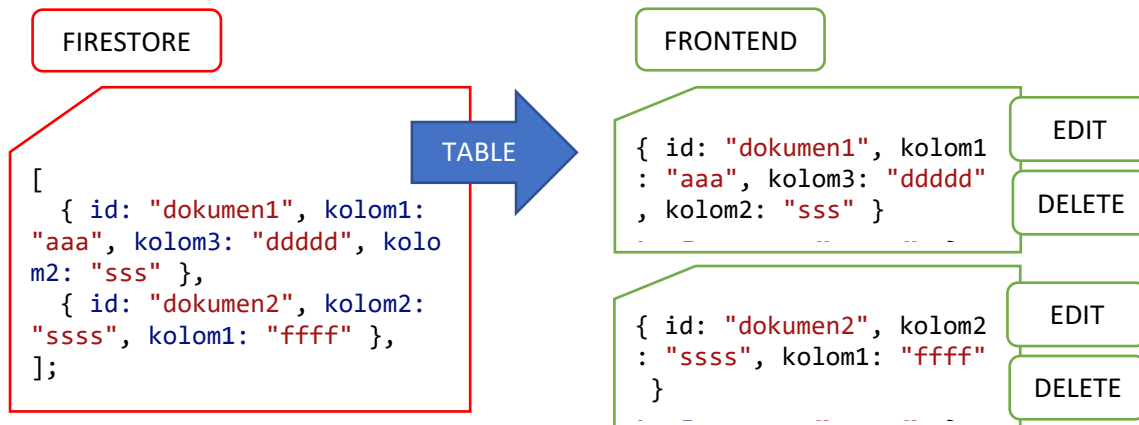
export default Latihan4;

```

6. MERAPIKAN FORM UNTUK TAMPILAN STATIS EDIT DAN DELETE

Action edit dan delete adalah action yang merusak data yang dipilih dari baris dalam table, untuk tampilan statis kita tampilkan data itu ke dalam console log

<https://github.com/edylee/fire-crud/blob/latihan5/admin-crud/src/Latihan5.js>



Latihan 5 Menampilkan Edit Data

Set Data :

Nama Collection :

Nama Dokumen :

Kolom1 :

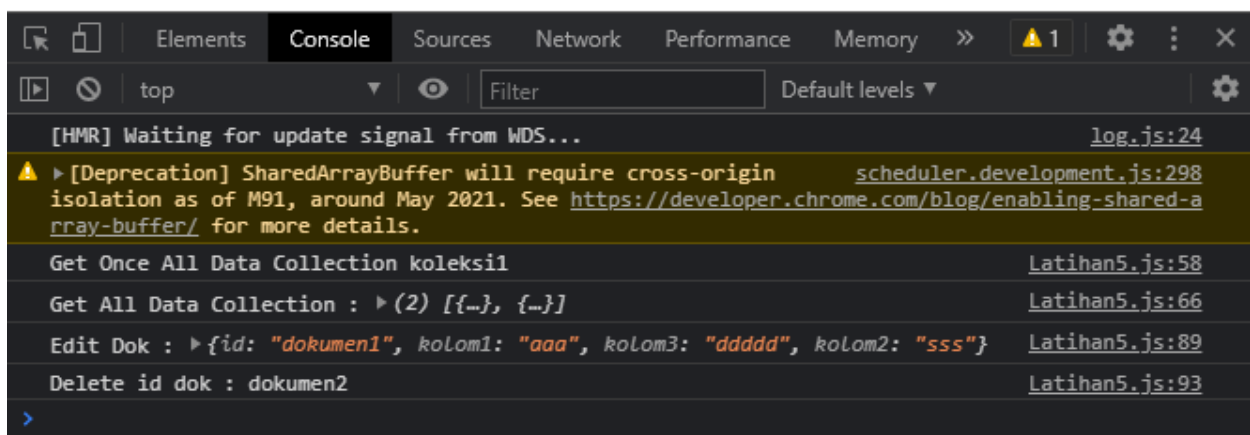
Kolom2 :

Data Firestore

```
[{"id":"dokumen1","kolom1":"aaa","kolom3":"dddd","kolom2":"sss"}, {"id":"dokumen2","kolom2":"ssss","kolom1":"ffff"}]
```

Table Firestore

kolom 1	kolom 2	kolom 3	EDIT	DELETE
aaa	sss	dddd	<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>
ffff	ssss		<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>




```

//src/Latihan5.js
import React, { useState } from "react";
//import firebase
import { db } from "../firebase";

function Latihan5() {
  //state
  const [nmCollection, setNmCollection] = useState("");
  const [nmDokumen, setNmDokumen] = useState("");
  const [kolom1, setKolom1] = useState("");
  const [kolom2, setKolom2] = useState("");

  //state untuk data dari hasil pembacaan
  const [dataCollection, setDataCollection] = useState([]);

  //button simpan fuction
  function onSimpanDokDgId() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    ;
    if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
    if (kolom1 === "") return console.log("Kolom1 Kosong");
    if (kolom2 === "") return console.log("Kolom2 Kosong");
    console.log(nmCollection, nmDokumen, kolom1, kolom2);
    db.collection(nmCollection)
      .doc(nmDokumen)
      .set({
        kolom1: kolom1,
        kolom2: kolom2,
      })
      .then(() => {
        console.log("Document successfully written!");
      })
      .catch((error) => {
        console.error("Error writing document: ", error);
      });
  }

  function onSimpanDok() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    ;
    if (kolom1 === "") return console.log("Kolom1 Kosong");
    if (kolom2 === "") return console.log("Kolom2 Kosong");
    console.log(nmCollection, kolom1, kolom2);
    // Add a new document in collection "cities"
    db.collection(nmCollection)
      .add({
        kolom1: kolom1,
        kolom2: kolom2,
      })
  }
}

```

```

        .then((doc) => {
            console.log("Added doc with ID : ", doc.id);
        })
        .catch((error) => {
            console.error("Error writing document: ", error);
        });
    }

    const getAllCol = () => {
        if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    };

    console.log("Get Once All Data Collection", nmCollection);
    db.collection(nmCollection)
        .get()
        .then((firecol) => {
            const data = firecol.docs.map((doc) => ({
                id: doc.id,
                ...doc.data(),
            }));
            console.log("Get All Data Collection :", data);
            setDataCollection(data);
        })
        .catch((error) => console.error("Error Get Data :", error));
    };

    const getDok = () => {
        if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    };

    if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
    console.log("Get Dokumen", nmCollection);
    db.collection(nmCollection)
        .doc(nmDokumen)
        .get()
        .then((doc) => {
            if (!doc.exists) return console.log("No such document!");
            return setDataCollection([{ id: doc.id, ...doc.data() }]);
        })
        .catch((error) => {
            console.log("Error getting document:", error);
        });
    };

    function onEditDok(row) {
        console.log("Edit Dok :", row);
    }

    function onDeleteDok(id_del) {
        console.log("Delete id dok :", id_del);
    }

```

```

return (
  <div>
    <h3>Latihan 5 Menampilkan Edit Data</h3>
    <fieldset>
      <legend>Set Data :</legend>
      <label htmlFor="nmCollection">Nama Collection :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="nmCollection"
        placeholder="Isi Nama Collection"
        onChange={(e) => setNmCollection(e.target.value)}
      />{ " " }
      <button onClick={getAllCol}>LIHAT COLL</button>
      <br></br>
      <br></br>
      <label htmlFor="nmDokumen">Nama Dokumen :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="nmDokumen"
        placeholder="Isi Kolom1"
        onChange={(e) => setNmDokumen(e.target.value)}
      />{ " " }
      <button onClick={getDok}>LIHAT DOK</button>
      <br></br>
      <br></br>
      <label htmlFor="kolom1">Kolom1 :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="kolom1"
        placeholder="Isi Kolom1"
        onChange={(e) => setKolom1(e.target.value)}
      />
      <br></br>
      <br></br>
      <label htmlFor="kolom2">Kolom2 :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="kolom2"
        placeholder="Isi Kolom2"
        onChange={(e) => setKolom2(e.target.value)}
      />
      <br></br>
      <br></br>
    </fieldset>
  </div>
)

```

```

    <button onClick={onSimpanDokDgId}>SIMPAN DOK DG ID</button>{"
"}
    <button onClick={onSimpanDok}>SIMPAN DOK</button>
</fieldset>
<fieldset>
    <legend>Data Firestore </legend>
    {JSON.stringify(dataCollection)}
</fieldset>
<fieldset>
    <legend>Table Firestore </legend>
    <table>
        <thead>
            <tr>
                <th>kolom 1</th>
                <th>kolom 2</th>
                <th>kolom 3</th>
                <th>EDIT</th>
                <th>DELETE</th>
            </tr>
        </thead>
        <tbody>
            {dataCollection?.map((row) => (
                <tr key={row.id} style={{ height: "2em" }}>
                    <td>{row.kolom1}</td>
                    <td>{row.kolom2}</td>
                    <td>{row.kolom3}</td>
                    <td>
                        <button onClick={() => onEditDok(row)}>EDIT</button>
                    </td>
                    <td>
                        <button onClick={() => onDeleteDok(row.id)}>DELETE</
button>
                    </td>
                </tr>
            ))}
        </tbody>
    </table>
</fieldset>
</div>
);
}

export default Latihan5;

```

Perubahan pada App.js

```

//src/App.js
// import Latihan1 from "./Latihan1";
//import Latihan2 from "./Latihan2";

```

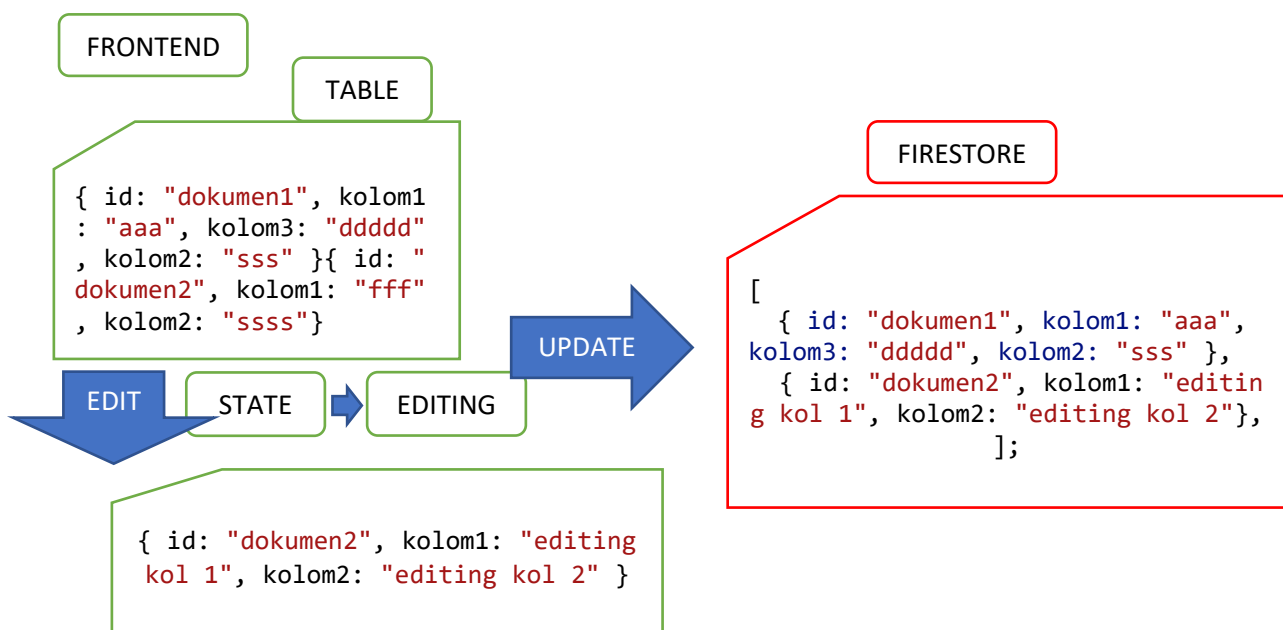
```

// import Latihan3 from "../Latihan3";
// import Latihan4 from "../Latihan4";
import Latihan5 from "../Latihan5";
function App() {
  return (
    <div>
      { /* <Latihan1 /> */ }
      { /* <Latihan2 /> */ }
      { /* <Latihan3 /> */ }
      { /* <Latihan4 /> */ }
      <Latihan5 />
    </div>
  );
}

export default App;

```

7. EDIT DATA



<https://firebase.google.com/docs/firestore/manage-data/add-data?authuser=0#update-data>

Untuk memudahkan pembacaan alur maka semua kitalakukan dalam satu halaman coding, nantinya bisa kita pisahkan dengan refactoring dan react context

- Membuat parameter2 state untuk editing
- Mengisi state dengan onclick dari tombol edit
- Menampilkan form editing
- Simpan data jika sudah di edit

<https://github.com/edycoleee/fire-crud/blob/latihan5a/admin-crud/src/Latihan5.js>

Latihan 5 Menampilkan Edit Data

Add Data :

Nama Collection : 1

Nama Dokumen :

Kolom1 :

Kolom2 :

Data Firestore

```
[{"id":"dokumen1","kolom1":"aaa","kolom3":"dddd","kolom2":"sss"}, {"id":"dokumen2"}
```

Table Firestore

kolom 1	kolom 2	kolom 3	EDIT	DELETE
aaa	sss	dddd	<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>
ffff	ssss		<input type="button" value="EDIT"/> 2	<input type="button" value="DELETE"/>

Edit Data :

Nama Collection : koleksi1

Nama Dokumen : dokumen2

Kolom1 :

Kolom2 : 3

1

```
[HMR] Waiting for update signal from WDS...
▶ [Deprecation] SharedArrayBuffer will require cross-origin isolation as of M91,
s://developer.chrome.com/blog/enabling-shared-array-buffer/ for more details.
Get Once All Data Collection koleksi1
Get All Data Collection : ▶ (2) [{-}, {-}]
Edit Dok : ▶ {id: "dokumen2", kolom1: "ffff", kolom2: "ssss"}
> |
```

setelah di edit

Latihan 5 Menampilkan Edit Data

Add Data :

Nama Collection : LIHAT COLL

Nama Dokumen : LIHAT DOK

Kolom1 :

Kolom2 :

SIMPAN DOK DG ID

SIMPAN DOK

Data Firestore

```
[{"id":"dokumen1","kolom3":"ddddd","kolom1":"aaa","kolom2":"sss"}, {"id":"dokumen2","kolom2":"editing kol 1","kolom1":"editing kol 1"}]
```

Table Firestore

kolom 1	kolom 2	kolom 3	EDIT	DELETE
aaa	sss	ddddd	<button>EDIT</button>	<button>DELETE</button>
editing kol 1	editing kol 1		<button>EDIT</button>	<button>DELETE</button>

Edit Data :

Nama Collection : koleksi1

Nama Dokumen : dokumen2

Kolom1 :

Kolom2 :

UPDATE

LIHAT COLL

```
//src/Latihan5.js
import React, { useState } from "react";
//import firebase
import { db } from "../firebase";

function Latihan5() {
  //state untuk add data
  const [nmCollection, setNmCollection] = useState("");
  const [nmDokumen, setNmDokumen] = useState("");
  const [kolom1, setKolom1] = useState("");
  const [kolom2, setKolom2] = useState("");

  //state untuk data dari hasil pembacaan
  const [dataCollection, setDataCollection] = useState([]);

  //state untuk edit data
  const [edtnmDokumen, setEdtNmDokumen] = useState("");
  const [edtkolom1, setEdtKolom1] = useState("");
  const [edtkolom2, setEdtKolom2] = useState("");

  //button simpan function simpan dokumen dg id (SET)
  function onSimpanDokDgId() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
    if (kolom1 === "") return console.log("Kolom1 Kosong");
    if (kolom2 === "") return console.log("Kolom2 Kosong");
    console.log(nmCollection, nmDokumen, kolom1, kolom2);
    db.collection(nmCollection)
      .doc(nmDokumen)
      .set({
```

```

        kolom1: kolom1,
        kolom2: kolom2,
    })
    .then(() => {
        console.log("Document successfully written!");
    })
    .catch((error) => {
        console.error("Error writing document: ", error);
    });
}

//button simpan fuction simpan dokumen auto id (ADD)
function onSimpanDok() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    if (kolom1 === "") return console.log("Kolom1 Kosong");
    if (kolom2 === "") return console.log("Kolom2 Kosong");
    console.log(nmCollection, kolom1, kolom2);
    db.collection(nmCollection)
        .add({
            kolom1: kolom1,
            kolom2: kolom2,
        })
        .then((doc) => {
            console.log("Added doc with ID : ", doc.id);
        })
        .catch((error) => {
            console.error("Error writing document: ", error);
        });
}

//button membaca semua data dari collection
const getAllCol = () => {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    console.log("Get Once All Data Collection", nmCollection);
    db.collection(nmCollection)
        .get()
        .then((firecol) => {
            const data = firecol.docs.map((doc) => ({
                id: doc.id,
                ...doc.data(),
            }));
            console.log("Get All Data Collection :", data);
            setDataCollection(data);
        })
        .catch((error) => console.error("Error Get Data :", error));
};

//button membaca semua data dari dokumen dalam collection
const getDok = () => {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
    console.log("Get Dokumen", nmCollection);
    db.collection(nmCollection)
        .doc(nmDokumen)
        .get()
        .then((doc) => {
            if (!doc.exists) return console.log("No such document!");
            return setDataCollection([
                { id: doc.id, ...doc.data() }
            ]);
        })
};

```



```

        .catch((error) => {
            console.log("Error getting document:", error);
        });
    });

//button mengambil data yang akan di edit
function onEditDok(row) {
    console.log("Edit Dok :", row);
    //memasukkan data kedalam state edit
    setEdtNmDokumen(row.id);
    setEdtKolom1(row.kolom1);
    setEdtKolom2(row.kolom2);
}

//button simpan edit
function onSimpanEdit() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    if (edtnmDokumen === "") return console.log("Nama Dokumen Kosong");
    if (edtkolom1 === "") return console.log("Kolom1 Kosong");
    if (edtkolom2 === "") return console.log("Kolom2 Kosong");
    console.log(nmCollection, edtnmDokumen, edtkolom1, edtkolom2);
    db.collection(nmCollection)
        .doc(edtnmDokumen)
        .update({
            kolom1: edtkolom1,
            kolom2: edtkolom2,
        })
        .then(() => {
            console.log("Updated doc ");
        })
        .catch((error) => {
            console.error("Error update document: ", error);
        });
}

//button utk delte data
function onDeleteDok(id_del) {
    console.log("Delete id dok :", id_del);
}

return (
    <div>
        <h3>Latihan 5 Menampilkan Edit Data</h3>
        { /* -----add data----- */ }
        <fieldset>
            <legend>Add Data :</legend>
            <label htmlFor="nmCollection">Nama Collection :</label>
            <input
                style={{ marginLeft: "1em" }}
                type="text"
                name="nmCollection"
                placeholder="Isi Nama Collection"
                onChange={(e) => setNmCollection(e.target.value)}
            />{ " " }
            <button onClick={getAllCol}>LIHAT COLL</button>
            <br></br>
            <br></br>
            <label htmlFor="nmDokumen">Nama Dokumen :</label>
            <input
                style={{ marginLeft: "1em" }}

```

```

        type="text"
        name="nmDokumen"
        placeholder="Isi Kolom1"
        onChange={(e) => setNmDokumen(e.target.value)}
      />{ " "}
      <button onClick={getDok}>LIHAT DOK</button>
      <br></br>
      <br></br>
      <label htmlFor="kolom1">Kolom1 :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="kolom1"
        placeholder="Isi Kolom1"
        onChange={(e) => setKolom1(e.target.value)}
      />
      <br></br>
      <br></br>
      <label htmlFor="kolom2">Kolom2 :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="kolom2"
        placeholder="Isi Kolom2"
        onChange={(e) => setKolom2(e.target.value)}
      />
      <br></br>
      <br></br>
      <button onClick={onSimpanDokDgId}>SIMPAN DOK DG ID</button>{ " "}
      <button onClick={onSimpanDok}>SIMPAN DOK</button>
    </fieldset>
    <fieldset>
      <legend>Data Firestore </legend>
      {JSON.stringify(dataCollection)}
    </fieldset>
    { /* -----table data----- */ }
    <fieldset>
      <legend>Table Firestore </legend>
      <table>
        <thead>
          <tr>
            <th>kolom 1</th>
            <th>kolom 2</th>
            <th>kolom 3</th>
            <th>EDIT</th>
            <th>DELETE</th>
          </tr>
        </thead>
        <tbody>
          {dataCollection?.map((row) => (
            <tr key={row.id} style={{ height: "2em" }}>
              <td>{row.kolom1}</td>
              <td>{row.kolom2}</td>
              <td>{row.kolom3}</td>
              <td>
                <button onClick={() => onEditDok(row)}>EDIT</button>
              </td>
              <td>
                <button onClick={() => onDeleteDok(row.id)}>DELETE</button>
              </td>
            </tr>
          )

```

```

        </td>
      </tr>
    )}
  </tbody>
</table>
</fieldset>
{ /* -----edit data----- */}
<fieldset>
  <legend>Edit Data :</legend>
  <p>Nama Collection : {nmCollection || "Kosong"}</p>
  <p>Nama Dokumen : {edtnmDokumen || "Kosong"}</p>
  <label htmlFor="kolom1">Kolom1 :</label>
  <input
    style={{ marginLeft: "1em" }}
    type="text"
    name="kolom1"
    value={edtkolom1 || ""}
    onChange={(e) => setEdtKolom1(e.target.value)}
  />
  <br></br>
  <br></br>
  <label htmlFor="kolom2">Kolom2 :</label>
  <input
    style={{ marginLeft: "1em" }}
    type="text"
    name="kolom2"
    value={edtkolom2 || ""}
    onChange={(e) => setEdtKolom2(e.target.value)}
  />
  <br></br>
  <br></br>
  <button onClick={onSimpanEdit}>UPDATE </button>{" "}
  <button onClick={getAllCol}>LIHAT COLL</button>
</fieldset>
</div>
);
}

export default Latihan5;

```

8. DELETE DATA DOKUMEN

Latihan 5 Menampilkan Edit Data

Add Data :

Nama Collection :

Nama Dokumen :

Kolom1 :

Kolom2 :

Data Firestore

```
[{"id":"dokumen1","kolom2":"aaa","kolom3":"ddddd","kolom1":"aaa"}, {"id":"yjRPAZT4sHbxxVzzimY2","kolom2":"delete2","kolom1":"delete1"}]
```

Table Firestore

kolom 1	kolom 2	kolom 3	EDIT	DELETE
aaa	aaa	ddddd	<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>
editing kol 1	editing kol 1		<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>
delete1	delete2		<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>

Console Log:

```
[HMR] Waiting for update signal from WDS...
[Deprecation] SharedArrayBuffer will require cross-origin isolation:
s://developer.chrome.com/blog/enabling-shared-array-buffer/ for more
Get Once All Data Collection koleksi1
Get All Data Collection : (3) [{"id": "dokumen1", "kolom1": "aaa", "kolom2": "aaa", "kolom3": "ddddd"}, {"id": "yjRPAZT4sHbxxVzzimY2", "kolom1": "delete1", "kolom2": "delete2", "kolom3": ""}]
```

<https://github.com/edycoleee/fire-crud/blob/latihan5b/admin-crud/src/Latihan5.js>

```
//src/Latihan5.js
import React, { useState } from "react";
//import firebase
import { db } from "../firebase";

function Latihan5() {
  //state untuk add data
  const [nmCollection, setNmCollection] = useState("");
  const [nmDokumen, setNmDokumen] = useState("");
  const [kolom1, setKolom1] = useState("");
  const [kolom2, setKolom2] = useState("");
```

```

//state untuk data dari hasil pembacaan
const [dataCollection, setDataCollection] = useState([]);

//state untuk edit data
const [edtnmDokumen, setEdtNmDokumen] = useState("");
const [edtkolom1, setEdtKolom1] = useState("");
const [edtkolom2, setEdtKolom2] = useState("");

//button simpan fuction simpan dokumen dg id (SET)
function onSimpanDokDgId() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
  if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
  if (kolom1 === "") return console.log("Kolom1 Kosong");
  if (kolom2 === "") return console.log("Kolom2 Kosong");
  console.log(nmCollection, nmDokumen, kolom1, kolom2);
  db.collection(nmCollection)
    .doc(nmDokumen)
    .set({
      kolom1: kolom1,
      kolom2: kolom2,
    })
    .then(() => {
      console.log("Document successfully written!");
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}

//button simpan fuction simpan dokumen auto id (ADD)
function onSimpanDok() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
  if (kolom1 === "") return console.log("Kolom1 Kosong");
  if (kolom2 === "") return console.log("Kolom2 Kosong");
  console.log(nmCollection, kolom1, kolom2);
  db.collection(nmCollection)
    .add({
      kolom1: kolom1,
      kolom2: kolom2,
    })
    .then((doc) => {
      console.log("Added doc with ID : ", doc.id);
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}

//button membaca semua data dari collection
const getAllCol = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
  console.log("Get Once All Data Collection", nmCollection);
  db.collection(nmCollection)
    .get()
    .then((firecol) => {
      const data = firecol.docs.map((doc) => ({
        id: doc.id,
        ...doc.data(),
      }));
    });
}

```

```

        console.log("Get All Data Collection :", data);
        setDataCollection(data);
    })
    .catch((error) => console.error("Error Get Data :", error));
};

//button membaca semua data dari dokumen dalam collection
const getDok = () => {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    if (nmDokumen === "") return console.log("Nama Dokumen Kosong");
    console.log("Get Dokumen", nmCollection);
    db.collection(nmCollection)
        .doc(nmDokumen)
        .get()
        .then((doc) => {
            if (!doc.exists) return console.log("No such document!");
            return setDataCollection([
                { id: doc.id, ...doc.data() }
            ]);
        })
        .catch((error) => {
            console.log("Error getting document:", error);
        });
};

//button mengambil data yang akan di edit
function onEditDok(row) {
    console.log("Edit Dok :", row);
    //memasukkan data kedalam state edit
    setEdtNmDokumen(row.id);
    setEdtKolom1(row.kolom1);
    setEdtKolom2(row.kolom2);
}

//button simpan edit
function onSimpanEdit() {
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    if (edtnmDokumen === "") return console.log("Nama Dokumen Kosong");
    if (edtkolom1 === "") return console.log("Kolom1 Kosong");
    if (edtkolom2 === "") return console.log("Kolom2 Kosong");
    console.log(nmCollection, edtnmDokumen, edtkolom1, edtkolom2);
    db.collection(nmCollection)
        .doc(edtnmDokumen)
        .update({
            kolom1: edtkolom1,
            kolom2: edtkolom2,
        })
        .then(() => {
            console.log("Updated doc ");
        })
        .catch((error) => {
            console.error("Error update document: ", error);
        });
}

//button utk delte data
function onDeleteDok(id_del) {
    console.log("Delete id dok :", id_del);
    if (nmCollection === "") return console.log("Nama Koleksi Kosong");
    if (!id_del) return console.log("Nama Dokumen Kosong");
    console.log(nmCollection, id_del);
    db.collection(nmCollection)

```

```

.doc(id_del)
.delete()
.then(() => {
  console.log("Deleted doc ");
})
.catch((error) => {
  console.error("Error update document: ", error);
});
}

return (
  <div>
    <h3>Latihan 5 Menampilkan Edit Data</h3>
    { /* -----add data----- */ }
    <fieldset>
      <legend>Add Data :</legend>
      <label htmlFor="nmCollection">Nama Collection :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="nmCollection"
        placeholder="Isi Nama Collection"
        onChange={(e) => setNmCollection(e.target.value)}
      />{ " " }
      <button onClick={getAllCol}>LIHAT COLL</button>
      <br></br>
      <br></br>
      <label htmlFor="nmDokumen">Nama Dokumen :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="nmDokumen"
        placeholder="Isi Kolom1"
        onChange={(e) => setNmDokumen(e.target.value)}
      />{ " " }
      <button onClick={getDok}>LIHAT DOK</button>
      <br></br>
      <br></br>
      <label htmlFor="kolom1">Kolom1 :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="kolom1"
        placeholder="Isi Kolom1"
        onChange={(e) => setKolom1(e.target.value)}
      />
      <br></br>
      <br></br>
      <label htmlFor="kolom2">Kolom2 :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="kolom2"
        placeholder="Isi Kolom2"
        onChange={(e) => setKolom2(e.target.value)}
      />
      <br></br>
      <br></br>
      <button onClick={onSimpanDokDgId}>SIMPAN DOK DG ID</button>{ " " }
    </div>
  )
)

```

```

    <button onClick={onSimpanDok}>SIMPAN DOK</button>
  </fieldset>
  <fieldset>
    <legend>Data Firestore </legend>
    {JSON.stringify(dataCollection)}
  </fieldset>
  { /* -----table data----- */ }
  <fieldset>
    <legend>Table Firestore </legend>
    <table>
      <thead>
        <tr>
          <th>kolom 1</th>
          <th>kolom 2</th>
          <th>kolom 3</th>
          <th>EDIT</th>
          <th>DELETE</th>
        </tr>
      </thead>
      <tbody>
        {dataCollection?.map((row) => (
          <tr key={row.id} style={{ height: "2em" }}>
            <td>{row.kolom1}</td>
            <td>{row.kolom2}</td>
            <td>{row.kolom3}</td>
            <td>
              <button onClick={() => onEditDok(row)}>EDIT</button>
            </td>
            <td>
              <button onClick={() => onDeleteDok(row.id)}>DELETE</button>
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </fieldset>
  { /* -----edit data----- */ }
  <fieldset>
    <legend>Edit Data </legend>
    <p>Nama Collection : {nmCollection || "Kosong"}</p>
    <p>Nama Dokumen : {edtnmDokumen || "Kosong"}</p>
    <label htmlFor="kolom1">Kolom1 </label>
    <input
      style={{ marginLeft: "1em" }}
      type="text"
      name="kolom1"
      value={edtkolom1 || ""}
      onChange={(e) => setEdtKolom1(e.target.value)}
    />
    <br><br>
    <label htmlFor="kolom2">Kolom2 </label>
    <input
      style={{ marginLeft: "1em" }}
      type="text"
      name="kolom2"
      value={edtkolom2 || ""}
      onChange={(e) => setEdtKolom2(e.target.value)}
    />
  </fieldset>

```



```

        <br></br>
        <br></br>
        <button onClick={onSimpanEdit}>UPDATE </button>{" "}
        <button onClick={getAllColl}>LIHAT COLL</button>
    </fieldset>
</div>
);
}

export default Latihan5;

```

8. DELETE KOLOM DI DALAM DATA DOKUMEN

Untuk menghapus kolom tertentu dari dokumen, gunakan metode `FieldValue.delete()` saat Anda memperbarui dokumen:

```

var cityRef = db.collection("cities").doc("BJ");

// Remove the 'capital' field from the document
var removeCapital = cityRef.update({
  capital: firebase.firestore.FieldValue.delete(),
});

```

perubahan file dalam latihan ini :

<https://github.com/edycoleee/fire-crud/blob/latihan6/admin-crud/src/Latihan6.js>

```

//src/App.js
// import Latihan1 from "../Latihan1";
//import Latihan2 from "../Latihan2";
// import Latihan3 from "../Latihan3";
// import Latihan4 from "../Latihan4";
//import Latihan5 from "../Latihan5";
import Latihan6 from "../Latihan6";
function App() {
  return (
    <div>
      {/* <Latihan1 /> */}
      {/* <Latihan2 /> */}
      {/* <Latihan3 /> */}
      {/* <Latihan4 /> */}
      {/* <Latihan5 /> */}
      <Latihan6 />
    </div>
  );
}

export default App;

```

```

//src/firebase.js
//import * as firebase from "firebase/app"; //before firebase v 8.00
import firebase from "firebase/app";
import "firebase/auth";
import "firebase/firestore";
import "firebase/storage";

```

```
const app = firebase.initializeApp({
  apiKey: "AIzaSyBFm3q7dmiKOIuNlkQy-4Zb9_LpQUYLVlI",
  authDomain: "coba-crud6.firebaseio.com",
  projectId: "coba-crud6",
  storageBucket: "coba-crud6.appspot.com",
  messagingSenderId: "863929184115",
  appId: "1:863929184115:web:3119e2d58b8d3dc71b5f06",
});

export const auth = app.auth();
export const db = app.firestore();
export const storage = firebase.storage();
export const Firebase = firebase;
```

Add Data :

Nama Collection :

Kolom1 :

Kolom2 :

Table Firestore

kolom 1	kolom 2	kolom 3	EDIT	DELETE
asd	sadsad		<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>
aaa	aaa	dddd	<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>
editing kol 1	editing kol 1		<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>
sss	dddd		<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>
delete1	delete2		<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>

Add / Delete Field Data inside dokumen CuWEfdecXEOFCNp6Rzz6

Kolom3 :

```
//src/Latihan6.js
import React, { useState } from "react";
//import firebase
import { db, Firebase } from "../firebase";

function Latihan6() {
  //state untuk add data
  const [nmCollection, setNmCollection] = useState("");
  const [kolom1, setKolom1] = useState("");
  const [kolom2, setKolom2] = useState("");
  const [kolom3, setKolom3] = useState("");

  //state untuk data dari hasil pembacaan
  const [dataCollection, setDataCollection] = useState([]);

  //state untuk edit data
  const [edtnmDokumen, setEdtNmDokumen] = useState("");
```

```

//button simpan fuction simpan dokumen auto id (ADD)
function onSimpanDok() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
  if (kolom1 === "") return console.log("Kolom1 Kosong");
  if (kolom2 === "") return console.log("Kolom2 Kosong");
  console.log(nmCollection, kolom1, kolom2);
  db.collection(nmCollection)
    .add({
      kolom1: kolom1,
      kolom2: kolom2,
    })
    .then((doc) => {
      getAllCol();
      setKolom1("");
      setKolom2("");
      console.log("Added doc with ID : ", doc.id);
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}

function onSimpanDokDgIdMerge() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
  if (edtnmDokumen === "") return console.log("Nama Dokumen Kosong");
  if (kolom3 === "") return console.log("Kolom3 Kosong");
  console.log(nmCollection, edtnmDokumen, kolom1, kolom2);
  db.collection(nmCollection)
    .doc(edtnmDokumen)
    .set(
      {
        kolom3: kolom3,
      },
      { merge: true }
    )
    .then(() => {
      getAllCol();
      console.log("Document successfully written!");
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}

//button membaca semua data dari collection
const getAllCol = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
  console.log("Get Once All Data Collection", nmCollection);
  db.collection(nmCollection)
    .get()
    .then((firecol) => {
      const data = firecol.docs.map((doc) => ({
        id: doc.id,
        ...doc.data(),
      }));
      console.log("Get All Data Collection :", data);
      setDataCollection(data);
    })
}

```

```

        .catch((error) => console.error("Error Get Data :", error));
    });

    //button mengambil data yang akan di edit
    function onEditDok(row) {
        console.log("Edit Dok :", row);
        //memasukkan data kedalam state edit
        setEdtNmDokumen(row.id);
    }

    //button utk delete data
    function onDeleteDok(id_del) {
        console.log("Delete id dok :", id_del);
    }

    //button utk delete field
    function onDeleteField() {
        console.log("Delete kolom3 dalam dok :", edtNmDokumen);
        if (nmCollection === "") return console.log("Nama Koleksi Kosong");
        if (!edtNmDokumen) return console.log("Nama Dokumen Kosong");
        console.log(nmCollection, edtNmDokumen);
        db.collection(nmCollection)
            .doc(edtNmDokumen)
            .update({
                kolom3: Firebase.firestore.FieldValue.delete(),
            })
            .then(() => {
                getAllCol();
                setKolom3("");
                console.log("Deleted doc ");
            })
            .catch((error) => {
                console.error("Error update document: ", error);
            });
    }

    return (
        <div>
            <h3>Latihan 6 Latihan Add Delete KOLom dalam Dok</h3>
            { /* -----add data----- */ }
            <fieldset>
                <legend>Add Data :</legend>
                <label htmlFor="nmCollection">Nama Collection :</label>
                <input
                    style={{ marginLeft: "1em" }}
                    type="text"
                    name="nmCollection"
                    placeholder="Isi Nama Collection"
                    onChange={(e) => setNmCollection(e.target.value)}
                />{ " " }
                <button onClick={getAllCol}>LIHAT COLL</button>
                <br><br>
                <label htmlFor="kolom1">Kolom1 :</label>
                <input
                    style={{ marginLeft: "1em" }}
                    type="text"
                    name="kolom1"
                    placeholder="Isi Kolom1"

```

```

        onChange={(e) => setKolom1(e.target.value)}
      />
      <br></br>
      <br></br>
      <label htmlFor="kolom2">Kolom2 :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="kolom2"
        placeholder="Isi Kolom2"
        onChange={(e) => setKolom2(e.target.value)}
      />
      <br></br>
      <br></br>
      <button onClick={onSimpanDok}>SIMPAN DOK</button>
    </fieldset>
    { /* -----table data----- */ }
    <fieldset>
      <legend>Table Firestore </legend>
      <table>
        <thead>
          <tr>
            <th>kolom 1</th>
            <th>kolom 2</th>
            <th>kolom 3</th>
            <th>EDIT</th>
            <th>DELETE</th>
          </tr>
        </thead>
        <tbody>
          {dataCollection?.map((row) => (
            <tr key={row.id} style={{ height: "2em" }}>
              <td>{row.kolom1}</td>
              <td>{row.kolom2}</td>
              <td>{row.kolom3}</td>
              <td>
                <button onClick={() => onEditDok(row)}>EDIT</button>
              </td>
              <td>
                <button onClick={() => onDeleteDok(row.id)}>DELETE</button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </fieldset>
    { /* -----Kolom data----- */ }
    <br></br>
    <fieldset>
      <legend>
        Add / Delete Field Data inside dokumen {edtnmDokumen || "Kosong"}{" "}
      </legend>
      <label htmlFor="kolom3">Kolom3 :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="kolom3"
        placeholder="Isi Kolom3"
        onChange={(e) => setKolom3(e.target.value)}
      />
    </fieldset>

```

```

        />
        <br></br>
        <br></br>
        <button onClick={onSimpanDokDgIdMerge}>ADD FIELD Kolom3</button>{" "}
        <button onClick={onDeleteField}>DELETE FIELD Kolom3</button>
    </fieldset>
</div>
);
}

export default Latihan6;

```

8. STEMPEL WAKTU SERVER

Anda dapat menetapkan kolom dalam dokumen ke stempel waktu server yang melacak kapan server menerima update. Terkadang jam di komputer client tidak sesuai dengan jam deserver, bias terjadi jika computer mengalami baterai bios lemah, ataupun ada client yang sengaja mengubah waktu di komputernya, maka kita bias menyimpan data ke server dengan jam server.

```

var docRef = db.collection("objects").doc("some-id");

// Update the timestamp field with the value from the server
var updateTimestamp = docRef.update({
  timestamp: firebase.firestore.FieldValue.serverTimestamp(),
});

```

<https://github.com/edyleeee/fire-crud/blob/latihan6a/admin-crud/src/Latihan6.js>

Latihan 6 Latihan Add Delete Kolom dalam Dok

Add Data :

Nama Collection : LIHAT COLL

Kolom1 :

Kolom2 :

Table Firestore

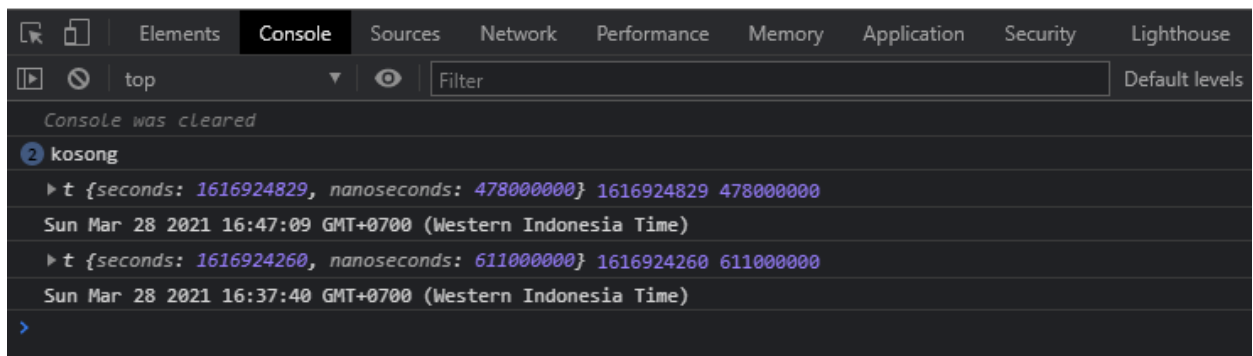
kolom 1	kolom 2	kolom 3	EDIT	DELETE	
asd	sadsad		<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>	{ "seconds": 1616924260, "nanoseconds": 611000000 } <input type="button" value="WAKTU"/>
aaa	aaa	dddd	<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>	{ "seconds": 1616924829, "nanoseconds": 478000000 } <input type="button" value="WAKTU"/>
editing kol 1	editing kol 1		<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>	<input type="button" value="WAKTU"/>
sss	dddd		<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>	<input type="button" value="WAKTU"/>
delete1	delete2		<input type="button" value="EDIT"/>	<input type="button" value="DELETE"/>	<input type="button" value="WAKTU"/>

Add / Delete Field Data inside dokumen Kosong

Kolom3 :

Data Firestore

```
[{"id": "CuWEfdecXEoFCNp6Rzz6", "kolom2": "sadsad", "kolom1": "asd", "createdAt": {"seconds": 1616924260, "nanoseconds": 611000000}, {"id": "o81nH1C3VJnPgqPUawfy", "kolom1": "sss", "kolom2": "dddd", "createdAt": {"seconds": 1616924829, "nanoseconds": 478000000}}, {"id": "yJRPAT4sHbx", "kolom1": "editing kol 1", "kolom2": "editing kol 1", "createdAt": {"seconds": 1616924829, "nanoseconds": 478000000}}]
```



```
//src/Latihan6.js
import React, { useState } from "react";
//import firebase
import { db, Firebase } from "../firebase";

function Latihan6() {
  //state untuk add data
  const [nmCollection, setNmCollection] = useState("");
  const [kolom1, setKolom1] = useState("");
  const [kolom2, setKolom2] = useState("");
  const [kolom3, setKolom3] = useState("");
}
```

```

//state untuk data dari hasil pembacaan
const [dataCollection, setDataCollection] = useState([]);

//state untuk edit data
const [edtnmDokumen, setEdtNmDokumen] = useState("");

//button simpan fuction simpan dokumen auto id (ADD)
function onSimpanDok() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
  if (kolom1 === "") return console.log("Kolom1 Kosong");
  if (kolom2 === "") return console.log("Kolom2 Kosong");
  console.log(nmCollection, kolom1, kolom2);
  db.collection(nmCollection)
    .add({
      kolom1: kolom1,
      kolom2: kolom2,
    })
    .then((doc) => {
      getAllCol();
      setKolom1("");
      setKolom2("");
      console.log("Added doc with ID : ", doc.id);
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}

function onSimpanDokDgIdMerge() {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
  if (edtnmDokumen === "") return console.log("Nama Dokumen Kosong");
  if (kolom3 === "") return console.log("Kolom3 Kosong");
  console.log(nmCollection, edtnmDokumen, kolom1, kolom2);
  db.collection(nmCollection)
    .doc(edtnmDokumen)
    .set(
      {
        kolom3: kolom3,
      },
      { merge: true }
    )
    .then(() => {
      getAllCol();
      console.log("Document successfully written!");
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}

//button membaca semua data dari collection
const getAllCol = () => {
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
  console.log("Get Once All Data Collection", nmCollection);
  db.collection(nmCollection)
    .get()
    .then((firecol) => {
      const data = firecol.docs.map((doc) => ({

```



```

        id: doc.id,
        ...doc.data(),
    }));
    console.log("Get All Data Collection :", data);
    setDataCollection(data);
  })
  .catch((error) => console.error("Error Get Data :", error));
};

//button mengambil data yang akan di edit
function onEditDok(row) {
  console.log("Edit Dok :", row);
  //memasukkan data kedalam state edit
  setEdtNmDokumen(row.id);
}

//button utk delete data
function onDeleteDok(id_del) {
  console.log("Delete id dok :", id_del);
}

//button utk delete field
function onDeleteField() {
  console.log("Delete kolom3 dalam dok :", edtnmDokumen);
  if (nmCollection === "") return console.log("Nama Koleksi Kosong");
  if (!edtnmDokumen) return console.log("Nama Dokumen Kosong");
  console.log(nmCollection, edtnmDokumen);
  db.collection(nmCollection)
    .doc(edtnmDokumen)
    .update({
      kolom3: Firebase.firestore.FieldValue.delete(),
    })
    .then(() => {
      getAllCol();
      setKolom3("");
      console.log("Deleted doc ");
    })
    .catch((error) => {
      console.error("Error update document: ", error);
    });
}

//button utk lihat jam server
function onAddTime() {
  db.collection(nmCollection)
    .doc(edtnmDokumen)
    .set(
      {
        createdAt: Firebase.firestore.FieldValue.serverTimestamp(),
      },
      { merge: true }
    )
    .then(() => {
      getAllCol();
      console.log("Document successfully written!");
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}

```

```

}

function onConversiTime(timestamp) {
  if (!timestamp) return console.log("kosong");
  console.log(timestamp, timestamp.seconds, timestamp.nanoseconds);
  const newDate = new Date(
    timestamp.seconds * 1000 + timestamp.nanoseconds / 1000000
  );
  return console.log(newDate);
}

return (
  <div>
    <h3>Latihan 6 Latihan Add Delete Kolom dalam Dok</h3>
    { /* -----add data----- */ }
    <fieldset>
      <legend>Add Data :</legend>
      <label htmlFor="nmCollection">Nama Collection :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="nmCollection"
        placeholder="Isi Nama Collection"
        onChange={(e) => setNmCollection(e.target.value)}
      />{ " " }
      <button onClick={getAllCol}>LIHAT COLL</button>
      <br></br>
      <br></br>
      <label htmlFor="kolom1">Kolom1 :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="kolom1"
        placeholder="Isi Kolom1"
        onChange={(e) => setKolom1(e.target.value)}
      />
      <br></br>
      <br></br>
      <label htmlFor="kolom2">Kolom2 :</label>
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="kolom2"
        placeholder="Isi Kolom2"
        onChange={(e) => setKolom2(e.target.value)}
      />
      <br></br>
      <br></br>
      <button onClick={onSimpanDok}>SIMPAN DOK</button>
    </fieldset>
    { /* -----table data----- */ }
    <fieldset>
      <legend>Table Firestore </legend>
      <table>
        <thead>
          <tr>
            <th>kolom 1</th>
            <th>kolom 2</th>
            <th>kolom 3</th>
            <th>EDIT</th>
          </tr>
        </thead>
      </table>
    </fieldset>
  </div>
)

```

```

        <th>DELETE</th>
      </tr>
    </thead>
    <tbody>
      {dataCollection?.map((row) => (
        <tr key={row.id} style={{ height: "2em" }}>
          <td>{row.kolom1}</td>
          <td>{row.kolom2}</td>
          <td>{row.kolom3}</td>
          <td>
            <button onClick={() => onEditDok(row)}>EDIT</button>
          </td>
          <td>
            <button onClick={() => onDeleteDok(row.id)}>DELETE</button>
          </td>
          <td>{JSON.stringify(row.createdAt)}</td>
          <td>
            <button onClick={() => onConversiTime(row.createdAt)}>
              WAKTU
            </button>
          </td>
        </tr>
      )})
    </tbody>
  </table>
</fieldset>
<!-- -----Kolom data----- */>
<br></br>
<fieldset>
  <legend>
    Add / Delete Field Data inside dokumen {edtnmDokumen || "Kosong"}{" "}
  </legend>
  <label htmlFor="kolom3">Kolom3 :</label>
  <input
    style={{ marginLeft: "1em" }}
    type="text"
    name="kolom3"
    placeholder="Isi Kolom3"
    onChange={(e) => setKolom3(e.target.value)}
  />
  <br></br>
  <br></br>
  <button onClick={onSimpanDokDgIdMerge}>ADD FIELD Kolom3</button>{" "}
  <button onClick={onDeleteField}>DELETE FIELD Kolom3</button>
  <br></br>
  <br></br>
  <button onClick={onAddTime}>Add Jam Server CreatedAt</button>
</fieldset>
<br></br>
<br></br>
<fieldset>
  <legend>Data Firestore </legend>
  {JSON.stringify(dataCollection)}
</fieldset>
</div>
  );
}

export default Latihan6;

```

9. KOLOM DENGAN KOLOM OBJECT BERTINGKAT

1. Jika dokumen Anda berisi objek bertingkat, Anda dapat menggunakan "notasi titik" untuk merujuk ke kolom bertingkat dalam dokumen saat memanggil `update()`:

```
// Create an initial document to update.
var frankDocRef = db.collection("users").doc("frank");
frankDocRef.set({
  name: "Frank",
  favorites: { food: "Pizza", color: "Blue", subject: "recess" },
  age: 12
});

// To update age and favorite color:
db.collection("users").doc("frank").update({
  "age": 13,
  "favorites.color": "Red"
})
.then(() => {
  console.log("Document successfully updated!");
});
```

2. Notasi titik memungkinkan Anda memperbarui satu kolom bertingkat tanpa menimpa kolom bertingkat lainnya. Jika Anda memperbarui kolom bertingkat tanpa notasi titik, Anda akan menimpa seluruh kolom peta, misalnya:

```
// Create our initial doc
db.collection("users").doc("frank").set({
  name: "Frank",
  favorites: {
    food: "Pizza",
    color: "Blue",
    subject: "Recess"
  },
  age: 12
}).then(function() {
  console.log("Frank created");
});

// Update the doc without using dot notation.
// Notice the map value for favorites.
db.collection("users").doc("frank").update({
  favorites: {
    food: "Ice Cream"
  }
}).then(function() {
  console.log("Frank food updated");
});

/*
Ending State, favorite.color and favorite.subject are no longer present:
/users
  /frank
  {
    name: "Frank",
    favorites: {
      food: "Ice Cream",
    },
    age: 12
  }
*/
```

```
*/
```

9. UPDATE ELEMEN DALAM ARRAY

Jika dokumen Anda berisi kolom array, Anda bisa menggunakan `arrayUnion()` dan `arrayRemove()` untuk menambah dan menghapus elemen. `arrayUnion()` menambahkan elemen ke array, tetapi hanya elemen yang belum ada. `arrayRemove()` menghapus semua instance dari setiap elemen yang diberikan.

```
var washingtonRef = db.collection("cities").doc("DC");

// Atomically add a new region to the "regions" array field.
washingtonRef.update({
  regions: firebase.firestore.FieldValue.arrayUnion("greater_virginia"),
});

// Atomically remove a region from the "regions" array field.
washingtonRef.update({
  regions: firebase.firestore.FieldValue.arrayRemove("east_coast"),
});
```

10. MENAMBAH NILAI NUMERIK

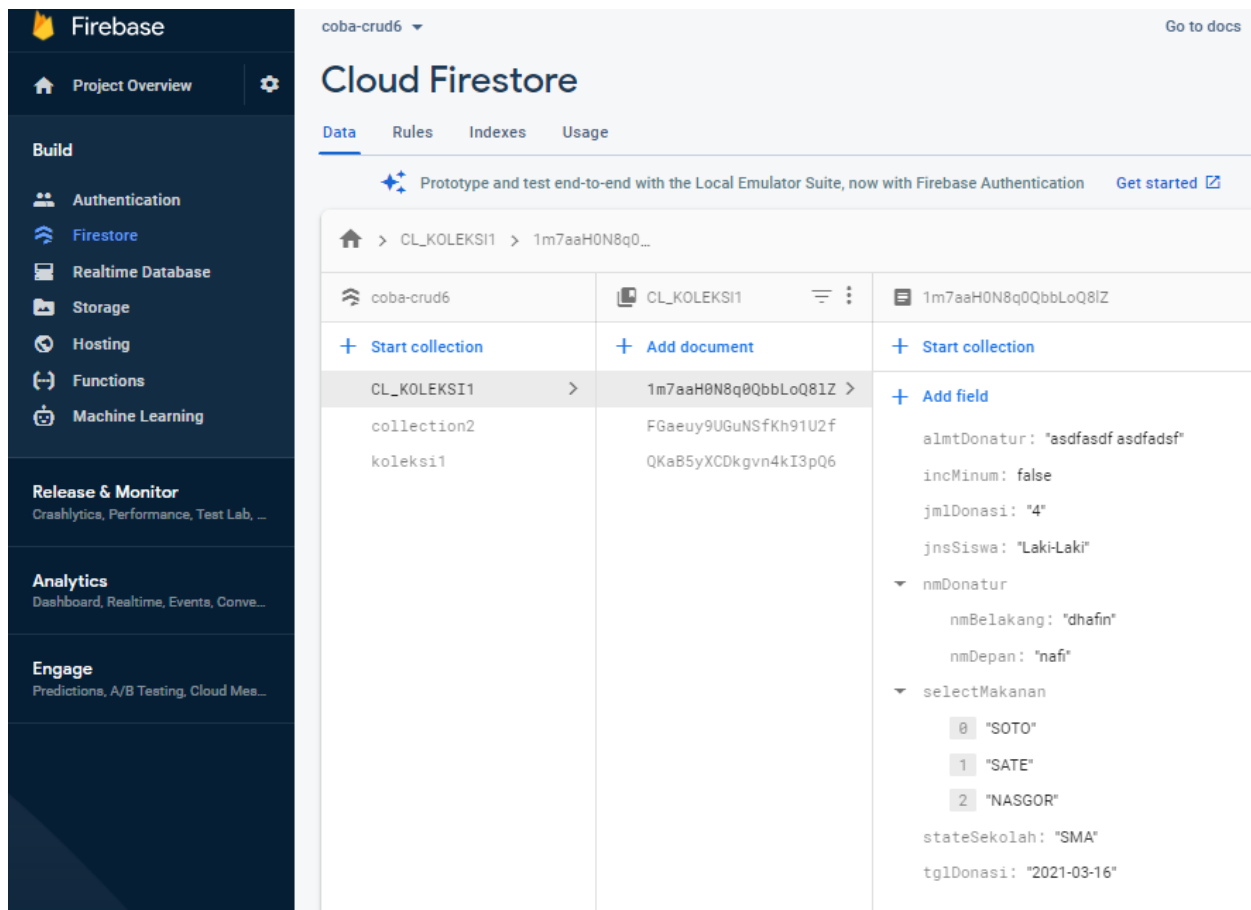
Anda bisa meningkatkan atau menurunkan nilai kolom numerik seperti yang ditunjukkan pada contoh berikut. Operasi peningkatan akan menambahkan atau mengurangi nilai kolom saat ini dengan jumlah tertentu. Jika kolom tidak ada atau jika nilai kolom saat ini bukan nilai numerik, operasi menetapkan kolom ke nilai yang diberikan.

```
var washingtonRef = db.collection("cities").doc("DC");

// Atomically increment the population of the city by 50.
washingtonRef.update({
  population: firebase.firestore.FieldValue.increment(50),
});
```

11. CRUD DATA

<https://github.com/edylee/fire-crud/blob/latihan7/admin-crud/src/Latihan7.js>



```
//bentuk data setelah disimpan
const DummyData = [
  {
    nmDonatur: {
      nmDepan: "",
      nmBelakang: "",
    },
    almtDonatur: "",
    jmlDonasi: 0,
    tglDonasi: "",
    selectMakanan: [],
    stateSekolah: "",
    incMinum: false,
    jnsSiswa: "",
  },
];
```

DISPLAY DATA

DATA DONASI KITA

TAMBAH BATAL SIMPAN

TABLE DATA

Nama	Alamat	Jumlah	Tgl	Makanan	Sekolah	Minuman	Jenis	EDT	DEL
nafi dhafin asdfasdf asdfasf	4	2021-03-16	SOTOSATENASGOR	SMA	TIDAK	Laki-Laki	<button>EDT</button>	<button>DEL</button>	
sss	sdsadsad	2	2021-03-29	SD	TIDAK		<button>EDT</button>	<button>DEL</button>	
dsfdsf dsf sdf dsfdsf	0	2021-03-29	SATENASGOR	S1	YA	Laki-Laki	<button>EDT</button>	<button>DEL</button>	

PERHATIAN

Data Telah Tersimpan x

TAMBAH DATA

DATA DONASI KITA

TAMBAH BATAL SIMPAN

TAMBAH DOKUMEN

Nama :

Alamat :

Jml Donasi : Tgl Donasi :

PILIH MAKANAN :
SATE
NASGOR
PIZZA

PILIH SEKOLAH :

Include Minuman : ☐ Air false

Jenis Siswa:

☐ Laki-Laki ☐ Wanita ☐ Semua

EDIT DATA

DATA DONASI KITA

TAMBAH BATAL SIMPAN

EDIT DOKUMEN

Nama :

Alamat :

Jml Donasi : Tgl Donasi :

PILIH MAKANAN :

PILIH SEKOLAH :

Include Minuman : ☐ Air false

Jenis Siswa: Laki-Laki

☐ Laki-Laki ☐ Wanita ☐ Semua

DELETE DATA

DATA DONASI KITA

TAMBAH BATAL SIMPAN

TABLE DATA

Nama	Alamat	Jumlah	Tgl	Makanan	Sekolah	Minuman	Jenis	EDT	DEL
nafi dhafin	asdfasdf asdfasdf	4	2021-03-16	SOTOSATENASGOR	SMA	TIDAK	Laki-Laki	EDT	DEL
dsfdsf dsf	sdf dsfdsf	0	2021-03-29	SATENASGOR	S1	YA	Laki-Laki	EDT	DEL

PERHATIAN

APAKAH DATA BENAR DIHAPUS ??? :QKaB5yXCDkgvn4kI3pQ6

```
import React, { useState, useEffect } from "react";
import { db, Firebase } from "../firebase";

const DEVELOP = true;
const NMCOLLECTION = "CL_KOLEKSI1";
const PILIHMAKANAN = ["SOTO", "SATE", "NASGOR", "PIZZA"];
const PILIHSEKOLAH = ["SD", "SMP", "SMA", "S1"];
function getCurrentDate(separator = "") {
  let newDate = new Date();
  let date = newDate.getDate();
  let month = newDate.getMonth() + 1;
  let year = newDate.getFullYear();

  return `${year}${separator}${
    month < 10 ? `0${month}` : `${month}`
  }`;
```



```

    }${separator}${date < 10 ? `0${date}` : `${date}`}`;
  }

function Latihan7() {
  const [nmDonatur, setNmDonatur] = useState({
    nmDepan: "",
    nmBelakang: "",
  });
  const [almtDonatur, setAlmtDonatur] = useState("");
  const [jmlDonasi, setJmlDonasi] = useState(0);
  const [tglDonasi, setTglDonasi] = useState(
    new Date().toISOString().slice(0, 10)
  );
  const [selectMakanan, setSelectMakanan] = useState([]);
  const [stateSekolah, setStateSekolah] = useState(PILIHSEKOLAH[0]);
  const [incMinum, setIncMinum] = useState(false);
  const [jnsSiswa, setJnsSiswa] = useState("");

  const [stTambah, setStTambah] = useState(false);
  const [stEdit, setStEdit] = useState(false);
  const [stLoading, setStLoading] = useState(false);
  const [stAlert, setStAlert] = useState("");
  const [dataCollection, setDataCollection] = useState([]);
  const [idEdtDok, setIdEdtDok] = useState("");
  const [stDelete, setStDelete] = useState(false);

  const handleChange = (e) => {
    let value = Array.from(e.target.selectedOptions, (option) => option.value);
    setSelectMakanan(value);
  };

  const pilihanMakanan = PILIHMAKANAN.map((item, index) => {
    return (
      <option key={index} value={item}>
        {item}
      </option>
    );
  });

  const pilihanSekolah = PILIHSEKOLAH.map((item, index) => {
    return (
      <option key={index} value={item}>
        {item}
      </option>
    );
  });

  const getAllCol = async () => {

```

```

await db
  .collection(NMCOLLECTION)
  .get()
  .then((firecol) => {
    const data = firecol.docs.map((doc) => ({
      id: doc.id,
      ...doc.data(),
    }));
    console.log("Get All Data Collection :", data);
    setDataCollection(data);
  })
  .catch((error) => console.error("Error Get Data :", error));
};

useEffect(() => {
  return getAllCol();
}, []);

const onTambah = () => {
  if (DEVELOP) console.log("onTambah");
  setStTambah(true);
  refreshData();
};

const onBatal = () => {
  if (DEVELOP) console.log("onBatal");
  setStTambah(false);
  setStEdit(false);
  refreshData();
};

const onSimpan = async () => {
  if (DEVELOP) console.log("onSimpan");
  if (nmDonatur.nmDepan === "") return setStAlert("Nama Depan Kosong");
  if (almtDonatur === "") return setStAlert("Alamat Kosong");
  if (jmlDonasi === 0) return setStAlert("Donasi Kosong");
  const newData = {
    nmDonatur,
    almtDonatur,
    jmlDonasi,
    tglDonasi,
    selectMakanan,
    stateSekolah,
    incMinum,
    jnsSiswa,
  };
  if (DEVELOP) console.log(newData);
  setStLoading(true);

```

```

if (stEdit) {
  if (DEVELOP) console.log("onSimpan Edit");
  await db
    .collection(NMCOLLECTION)
    .doc(idEdtDok)
    .update(newData)
    .then(() => {
      console.log("Updated doc ");
    })
    .catch((error) => {
      console.error("Error update document: ", error);
    });
} else {
  if (DEVELOP) console.log("onSimpan Tambah");

  await db
    .collection(NMCOLLECTION)
    .add(newData)
    .then((doc) => {
      console.log("Added doc with ID : ", doc.id);
    })
    .catch((error) => {
      console.error("Error writing document: ", error);
    });
}
await getAllCol();
setStLoading(false);
setStTambah(false);
setStEdit(false);
refreshData();
setStAlert("Data Telah Tersimpan");
};

const onEdit = (data) => {
  if (DEVELOP) console.log("onEdit", data);
  setNmDonatur({
    nmDepan: data.nmDonatur.nmDepan,
    nmBelakang: data.nmDonatur.nmBelakang,
  });
  setIdEdtDok(data.id);
  setAlmtDonatur(data.almtDonatur);
  setJmlDonasi(data.jmlDonasi);
  setTglDonasi(data.tglDonasi);
  setSelectMakanan(data.selectMakanan);
  setStateSekolah(data.stateSekolah);
  setIncMinum(data.incMinum);
  setJnsSiswa(data.jnsSiswa);
  setStAlert("");
  setStTambah(true);

```

```

    setStEdit(true);
};

const refreshData = () => {
    setNmDonatur({
        nmDepan: "",
        nmBelakang: "",
    });
    setAlmtDonatur("");
    setJmlDonasi(0);
    setTglDonasi(getCurrentDate("-"));
    setSelectMakanan([]);
    setStateSekolah(PILIHSEKOLAH[0]);
    setIncMinum(false);
    setJnsSiswa("");
    setStAlert("");
    setIdEdtDok("");
};

const onDelete = async () => {
    setStLoading(true);
    await db
        .collection(NMCOLLECTION)
        .doc(idEdtDok)
        .delete()
        .then(() => {
            console.log("Deleted Data");
        })
        .catch((error) => {
            console.error("Error writing document: ", error);
        });
    await getAllCol();
    setStLoading(false);
    setStDelete(false);
    setStAlert("Data Telah Terhapus : ", idEdtDok);
};

const onStateDelete = (idDok) => {
    setIdEdtDok(idDok);
    setStDelete(true);
    setStAlert("APAKAH DATA BENAR DIHAPUS ??? : " + idDok + " ");
};

const onClearAlert = () => {
    setStAlert("");
    setStDelete(false);
};

useEffect(() => {

```

```

const setAsyncTimeout = (cb, timeout = 0) =>
  new Promise((resolve) => {
    setTimeout(() => {
      cb();
      resolve();
    }, timeout);
  });

const ClearAlert = async () => {
  await setAsyncTimeout(() => {
    onClearAlert();
  }, 5000);
};

if (stAlert !== "") return ClearAlert();
}, [stAlert]);

if (stLoading) return <>Loading...</>;

return (
  <div style={{ marginLeft: "3em", maxWidth: "800px" }}>
    <h3>DATA DONASI KITA</h3>
    <br></br>
    <button onClick={() => onTambah()} disabled={stTambah}>
      TAMBAH
    </button>{" "}
    <button onClick={() => onBatal()} disabled={!stTambah}>
      BATAL
    </button>{" "}
    <button onClick={() => onSimpan()} disabled={!stTambah}>
      SIMPAN
    </button>
    <br></br>
    <br></br>
    {stTambah || stEdit ? (
      <fieldset>
        {stEdit ? (
          <legend>EDIT DOKUMEN</legend>
        ) : (
          <legend>TAMBAH DOKUMEN</legend>
        )}
        <label htmlFor="kolom1">Nama :</label>
        <input
          style={{ marginLeft: "1em" }}
          type="text"
          name="kolom1"
          placeholder="Nama Depan"
          onChange={(e) =>
            setNmDonatur({ ...nmDonatur, nmDepan: e.target.value })
          }
        >
      </fieldset>
    ) : null}
  </div>
);

```

```

    }
    value={nmDonatur?.nmDepan || ""}
  />
  <input
    style={{ marginLeft: "1em" }}
    type="text"
    name="kolom1"
    placeholder="Nama Belakang"
    onChange={(e) =>
      setNmDonatur({ ...nmDonatur, nmBelakang: e.target.value
    })

    }
    value={nmDonatur?.nmBelakang || ""}
  />
  <br></br>
  <br></br>
  <label htmlFor="alamat">Alamat :</label>
  <textarea
    id="alamat"
    name="alamat"
    rows="2"
    cols="50"
    placeholder="Alamat"
    onChange={(e) => setAlmtDonatur(e.target.value)}
    value={almtDonatur || ""}
  />
  <br></br>
  <br></br>
  <label htmlFor="kolom1">Jml Donasi :</label>
  <input
    style={{ marginLeft: "1em" }}
    type="number"
    name="kolom1"
    placeholder="Isi Kolom1"
    onChange={(e) => setJmlDonasi(e.target.value)}
    value={jmlDonasi || 0}
  />{" "}
  <label htmlFor="kolom1">Tgl Donasi :</label>
  <input
    style={{ marginLeft: "1em" }}
    type="date"
    name="kolom1"
    placeholder="Isi Kolom1"
    onChange={(e) => setTglDonasi(e.target.value)}
    value={tglDonasi}
    //value={stEdit ? tglDonasi : new Date().toISOString().slice(0, 10)}
  />
  <br></br>

```

```

<br></br>
<label>PILIH MAKANAN : </label>
<select
  id="select1"
  style={{ width: "150px" }}
  multiple={true}
  onChange={handleChange}
  value={selectMakanan}
>
  {pilihanMakanan}
</select>
<br></br>
<label>PILIH SEKOLAH : {" "}</label>
<select
  id="plhSekolah"
  style={{ width: "150px" }}
  onChange={(e) => setStateSekolah(e.target.value)}
  value={stateSekolah}
>
  {pilihanSekolah}
</select>
<br></br>
<label>Include Minuman : {" "}</label>
<input
  type="checkbox"
  id="minuman"
  name="minuman"
  onChange={(e) => setIncMinum(e.target.checked)}
  checked={incMinum}
/>
<label htmlFor="vehicle1"> Air</label> {JSON.stringify(incMi
num)}}
<br></br>
<label>Jenis Siswa: {jnsSiswa || ""}</label>
<div onChange={(e) => setJnsSiswa(e.target.value)}>
  <input type="radio" id="male" name="gender" value="Laki-
Laki" />
  <label htmlFor="male">Laki-Laki</label>
  <input type="radio" id="female" name="gender" value="Wanita" />
  <label htmlFor="female">Wanita</label>
  <input type="radio" id="semua" name="gender" value="Semua"
/>
  <label htmlFor="female">Semua</label>
</div>
</fieldset>
) : (
<fieldset>
  <legend>TABLE DATA</legend>

```

```

<table>
  <thead>
    <tr>
      <th>Nama</th>
      <th>Alamat</th>
      <th>Jumlah</th>
      <th>Tgl</th>
      <th>Makanan</th>
      <th>Sekolah</th>
      <th>Minuman</th>
      <th>Jenis</th>
      <th>EDT</th>
      <th>DEL</th>
    </tr>
  </thead>
  <tbody>
    {dataCollection?.map((row) => (
      <tr key={row.id} style={{ height: "2em" }}>
        <td>
          {row.nmDonatur.nmDepan} {row.nmDonatur.nmBelakang}
        </td>
        <td>{row.almtDonatur}</td>
        <td>{row.jmlDonasi}</td>
        <td>{row.tglDonasi}</td>
        <td>{row.selectMakanan}</td>
        <td>{row.stateSekolah}</td>
        <td>{row.incMinum ? "YA" : "TIDAK"}</td>
        <td>{row.jnsSiswa}</td>
        <td>
          <button onClick={() => onEdit(row)}>EDT</button>
        </td>
        <td>
          <button
            onClick={() => onStateDelete(row.id)}
            disabled={stDelete}
          >
            DEL
          </button>
        </td>
      </tr>
    ))}
  </tbody>
</table>
</fieldset>
))
<br></br>
{stAlert !== "" ? (
  <fieldset style={{ backgroundColor: "#F8DFF1" }}>
    <legend>PERHATIAN</legend>

```



```

        {stAlert}{" "}
        <button
          onClick={() => onClearAlert()}
          style={{ backgroundColor: "red" }}
        >
          x
        </button>{" "}
        {stDelete} && <button onClick={() => onDelete()}>DEL</button>
      }
    </fieldset>
  ) : (
    ""
  )
}
{JSON.stringify(tglDonasi)}
<br></br>
</div>
);
}

export default Latihan7;

```

14. MELAKUKAN QUERY SEDERHANA

DATA DONASI KITA

TAMBAH BATAL SIMPAN

TABLE DATA

Nama	Alamat	Jumlah	Tgl	Makanan	Sekolah	Minuman	Jenis	EDT	DEL
nafi dhafin	semarang	0	2021-03-16	SOTOSATENASGOR	SMA	TIDAK	Laki-Laki	EDT	DEL
silmi aira	tlogosari	1000	2021-04-04	NASGORPIZZA	SD	YA	Wanita	EDT	DEL
dika aa	demak	2	2021-03-30	SOTOSATE	SMA	TIDAK	Laki-Laki	EDT	DEL
wahid amrullah	semarang	10	2021-04-05	NASGORPIZZA	S1	YA	Laki-Laki	EDT	DEL
sdfdsf sdfdsf	sdewrewr	5	2021-04-05	SATE	SMP	YA	Laki-Laki	EDT	DEL

Filter Data

CARI DATA :

Pada cloud firestore query bisa dilakukan tanpa melalui indexing, tapi saat mensort data akan membutuhkan index composite, baiknya untuk mengurutkan data menggunakan operasi sort array, sehingga mempercepat proses.

<https://github.com/edycoleeee/fire-crud/blob/latihan8/admin-crud/src/Latihan8.js>

```

import React, { useState } from "react";
import { db } from "../firebase";

const PILIHCARI = ["Nama", "Alamat", "Tanggal", "Makanan", "Sekolah"];
const NMCOLLECTION = "CL_KOLEKSI1";
const PILIHSEKOLAH = ["SD", "SMP", "SMA", "S1"];

function Latihan8({ setStAlert, setDataCollection }) {
  const [statePlhCari, setStatePlhCari] = useState(PILIHCARI[0]);
  const [stCari, setStCari] = useState("");

  const [stateSekolah, setStateSekolah] = useState(PILIHSEKOLAH[0]);
  const pilihanSekolah = PILIHSEKOLAH.map((item, index) => {
    return (
      <option key={index} value={item}>
        {item}
      </option>
    );
  });

  const pilihanCari = PILIHCARI.map((item, index) => {
    return (
      <option key={index} value={item}>
        {item}
      </option>
    );
  });

  function onPilih(e) {
    setStatePlhCari(e.target.value);
    console.log(e.target.value);
    if (e.target.value === "Tanggal") {
      setStCari(new Date().toISOString().slice(0, 10));
    } else {
      setStCari("");
    }
  }

  async function onCari() {
    let fieldCari = "";
    const getCari = async (Cari) => {
      await db
        .collection(NMCOLLECTION)
        .where(fieldCari, "==", Cari)
        .get()
        .then((firecol) => {
          const data = firecol.docs.map((doc) => ({
            id: doc.id,
            ...doc.data(),
          }));
        });
    };
  }
}

```

```

    }));
    console.log("Get All Data Collection :", data);
    setDataCollection(data);
  })
  .catch((error) => console.error("Error Get Data :", error));

if (statePlhCari === "Sekolah") {
  fieldCari = "stateSekolah";
  return await getCari(stateSekolah);
}

if (stCari === "") return setStAlert("Pencarian Kosong");
console.log(stCari);

if (statePlhCari === "Makanan") {
  return await db
    .collection(NMCOLLECTION)
    .where("selectMakanan", "array-contains", stCari)
    .get()
    .then((firecol) => {
      const data = firecol.docs.map((doc) => ({
        id: doc.id,
        ...doc.data(),
      }));
      console.log("Get All Data Collection :", data);
      setDataCollection(data);
      if (data.length === 0) setStAlert("Pencarian Kosong");
    })
    .catch((error) => console.error("Error Get Data :", error));
}

if (statePlhCari === "Nama") fieldCari = "nmDonatur.nmDepan";
if (statePlhCari === "Alamat") fieldCari = "almtDonatur";
if (statePlhCari === "Tanggal") fieldCari = "tglDonasi";
return await getCari(stCari);
}

return (
  <div>
    <fieldset>
      <legend>Filter Data</legend>
      <label>CARI DATA : </label>
      <select
        id="plhSekolah"
        style={{ width: "150px" }}
        onChange={(e) => onPilih(e)}
        value={statePlhCari}
      >
        {pilihanCari}
      </select>
    </fieldset>
  </div>
);

```

```

    </select>
    {statePlhCari === "Sekolah" ? (
      <select
        style={{ width: "150px", marginLeft: "1em" }}
        onChange={(e) => setStateSekolah(e.target.value)}
        value={stateSekolah}
      >
        {pilihanSekolah}
      </select>
    ) : (
      <input
        style={{ marginLeft: "1em" }}
        type="text"
        name="kolom1"
        placeholder="Cari Data"
        onChange={(e) => setStCari(e.target.value)}
        value={stCari || ""}
      />
    )}{" "}
    <button onClick={() => onCari()}>CARI</button>
  </fieldset>
</div>
);
}

export default Latihan8;

```

penambahan di latihan 7

```

import Latihan8 from "./Latihan8";
.....
<Latihan8 setStAlert={setStAlert} setDataCollection={setDataCollection}
/>
.....

```

Contoh MockUp Data

Data contoh

Untuk memulai, tulis beberapa data mengenai kota agar kita dapat melihat berbagai cara untuk membacanya kembali:

```

var citiesRef = db.collection("cities");

citiesRef.doc("SF").set({
  name: "San Francisco",
  state: "CA",
  country: "USA",
  capital: false,
  population: 860000,

```

```

    regions: ["west_coast", "norcal"],
  });
citiesRef.doc("LA").set({
  name: "Los Angeles",
  state: "CA",
  country: "USA",
  capital: false,
  population: 3900000,
  regions: ["west_coast", "socal"],
});
citiesRef.doc("DC").set({
  name: "Washington, D.C.",
  state: null,
  country: "USA",
  capital: true,
  population: 680000,
  regions: ["east_coast"],
});
citiesRef.doc("TOK").set({
  name: "Tokyo",
  state: null,
  country: "Japan",
  capital: true,
  population: 9000000,
  regions: ["kanto", "honshu"],
});
citiesRef.doc("BJ").set({
  name: "Beijing",
  state: null,
  country: "China",
  capital: true,
  population: 21500000,
  regions: ["jingjinji", "hebei"],
});

```

Mendapatkan beberapa dokumen dari koleksi

Anda juga dapat mengambil beberapa dokumen dengan 1 permintaan, dengan membuat kueri dokumen dalam koleksi. Misalnya, Anda dapat menggunakan `where()` guna membuat kueri untuk semua dokumen yang memenuhi kondisi tertentu, kemudian menggunakan `get()` untuk mengambil hasilnya:

```

db.collection("cities")
  .where("capital", "==", true)
  .get()
  .then((querySnapshot) => {
    querySnapshot.forEach((doc) => {
      // doc.data() is never undefined for query doc snapshots
      console.log(doc.id, " => ", doc.data());
    });
  })
  .catch((error) => {
    console.log("Error getting documents: ", error);
  });

```

15. MELAKUKAN QUERY GABUNGAN

Cloud Firestore menyediakan fungsi kueri yang dapat diandalkan untuk menentukan dokumen mana yang ingin Anda ambil dari koleksi atau grup koleksi. Kueri ini juga dapat digunakan dengan `get()` atau `addSnapshotListener()`, seperti yang dijelaskan dalam [Mendapatkan Data](#) dan [Mendapatkan Update Realtime](#).

```
//Kueri sederhana
//Kueri berikut menampilkan semua kota dengan negara bagian CA:
// Create a reference to the cities collection
var citiesRef = db.collection("cities");

// Create a query against the collection.
var query = citiesRef.where("state", "==", "CA");

//Kueri berikut menampilkan semua ibu kota:
var citiesRef = db.collection("cities");

var query = citiesRef.where("capital", "==", true);

//Menjalankan kueri
db.collection("cities").where("capital", "==", true)
    .get()
    .then((querySnapshot) => {
        querySnapshot.forEach((doc) => {
            // doc.data() is never undefined for query doc snapshots
            console.log(doc.id, " => ", doc.data());
        });
    })
    .catch((error) => {
        console.log("Error getting documents: ", error);
    });
```

Operator kueri

Metode `where()` mengambil tiga parameter: kolom untuk difilter, operator perbandingan, dan nilai. Cloud Firestore mendukung operator perbandingan berikut ini:

- `<` kurang dari
- `<=` kurang dari atau sama dengan
- `==` sama dengan
- `>` lebih dari
- `>=` lebih dari atau sama dengan
- [!= tidak sama dengan](#)
- [array-contains](#)
- [array-contains-any](#)
- [in](#)
- [not-in](#)

```
//Contoh
citiesRef.where("state", "==", "CA");
citiesRef.where("population", "<", 100000);
```

```
citiesRef.where("name", ">=", "San Francisco");

//Tidak sama dengan (!=)
citiesRef.where("capital", "!=", false);
```

Kueri ini menampilkan setiap dokumen `city` dengan kolom `capital` yang sudah ada dengan nilai selain `false` atau `null`. Ini mencakup dokumen `city` dengan nilai kolom `capital` sama dengan `true` atau nilai non-boolean berapa pun selain `null`. Kueri ini tidak menampilkan dokumen `city` yang tidak memiliki kolom `capital`. **Kueri tidak sama dengan (`!=`) dan `not-in` mengecualikan dokumen yang tidak memiliki kolom tertentu.**

Kolom sudah ada saat ditetapkan ke nilai apa pun, termasuk string kosong (`""`), `null`, dan `NaN` (bukan angka). Perhatikan bahwa nilai kolom `null` tidak cocok dengan klausa `!=`, karena `x != null` bernilai `undefined`.

Peringatan: Klausa kueri `!=` mungkin cocok dengan banyak dokumen dalam koleksi. Untuk mengontrol jumlah atau hasil yang ditampilkan, gunakan klausu batas atau beri nomor kueri.

Batasan

Perhatikan batasan berikut untuk kueri `!=`:

- Hanya dokumen yang sudah memiliki kolom tertentu yang dapat dicocokkan dengan kueri.
- Anda tidak dapat menggabungkan `not-in` dan `!=` dalam kueri gabungan.
- Pada kueri gabungan, perbandingan rentang (`<`, `<=`, `>`, `>=`) dan tidak sama dengan (`!=`, `not-in`) harus memiliki semua filter di kolom yang sama.

Anda dapat menggunakan operator `array-contains` untuk memfilter berdasarkan nilai array. Contoh:

```
//Keanggotaan array
```

```
citiesRef.where("regions", "array-contains", "west_coast");
```

Kueri ini menampilkan setiap dokumen `city`, dengan kolom `regions` yang berupa array berisi `west_coast`. Jika array memiliki beberapa instance dari nilai yang Anda buat kuerinya, dokumen dimasukkan dalam hasil hanya sekali.

Anda dapat menggunakan maksimum satu klausa `array-contains` per kueri. Anda tidak dapat menggabungkan `array-contains` dengan `array-contains-any`.

`in`, `not-in`, dan `array-contains-any`

Gunakan operator `in` untuk menggabungkan hingga 10 klausa kesetaraan (`==`) di kolom yang sama dengan `OR` yang logis. Kueri `in` menampilkan dokumen dengan kolom tertentu yang cocok dengan salah satu nilai perbandingan. Contoh:

```
//in, not-in, dan array-contains-any
```

```
citiesRef.where("country", "in", ["USA", "Japan"]);
```

`not-in`

Gunakan operator `not-in` untuk menggabungkan hingga 10 klausa tidak sama dengan (`!=`) di kolom yang sama dengan `AND` yang logis. Kueri `not-in` menampilkan dokumen yang memiliki kolom tertentu, bukan `null`, dan tidak cocok dengan nilai perbandingan apa pun. Contoh:

```
//not-in
```

```
citiesRef.where("country", "not-in", ["USA", "Japan"]);
```

Kueri ini menampilkan setiap dokumen `city` yang memiliki kolom `country` dan tidak ditetapkan ke `USA`, `Japan`, atau `null`. Dari contoh data, ini mencakup dokumen `London` dan `Hong Kong`.

Kueri `not-in` mengecualikan dokumen yang tidak memiliki kolom tertentu. Kolom ada jika disetel ke nilai apa pun, termasuk string kosong (`""`), `null`, dan `NaN` (bukan angka). Perhatikan bahwa `x != null` bernilai `undefined`. Kueri `not-in` dengan `null` sebagai salah satu dari nilai perbandingan tidak cocok dengan dokumen mana pun.

`array-contains-any`

Gunakan operator `array-contains-any` untuk menggabungkan hingga 10 klausa `array-contains` di kolom yang sama dengan `OR` yang logis. Kueri `array-contains-any` menampilkan dokumen dengan kolom tertentu yang berupa array berisi satu atau beberapa nilai perbandingan:

```
//array-contains-any
citiesRef.where("regions", "array-contains-any", ["west_coast", "east_coast"]);
```

Kueri ini menampilkan setiap dokumen kota dengan kolom `region` yang berupa array berisi `west_coast` atau `east_coast`. Dari contoh data, ini mencakup dokumen `SF`, `LA`, dan `DC`.

Hasil dari `array-contains-any` akan dihapus duplikatnya. Meskipun kolom array dokumen cocok dengan lebih dari satu nilai perbandingan, kumpulan hasilnya hanya mencakup dokumen tersebut sekali.

`array-contains-any` selalu memfilter berdasarkan jenis data array. Misalnya, kueri di atas tidak akan menampilkan dokumen kota jika kolom `region` bukan berupa array, melainkan string `west_coast`.

Anda dapat menggunakan nilai array sebagai nilai perbandingan untuk `in`, tetapi tidak seperti `array-contains-any`, klausa cocok dengan kecocokan persis panjang array, urutan, dan nilai. Contoh:

```
citiesRef.where("region", "in", ["west_coast", "east_coast"]);
```

Kueri ini menampilkan setiap dokumen kota dengan kolom `region` yang berupa array berisi satu elemen `west_coast` atau `east_coast`. Dari contoh data, hanya dokumen `DC` yang memenuhi syarat dengan kolom `region` dari `["east_coast"]`. Namun, dokumen `SF` tidak cocok karena kolom `region`-nya adalah `["west_coast", "norcal"]`.

Batasan

Perhatikan batasan berikut untuk `in`, `not-in`, dan `array-contains-any`:

- `in`, `not-in`, dan `array-contains-any` mendukung hingga 10 nilai perbandingan.
- Anda dapat menggunakan maksimum satu klausa `array-contains` per kueri. Anda tidak dapat menggabungkan `array-contains` dengan `array-contains-any`.
- Anda dapat menggunakan maksimum satu klausa `in`, `not-in`, atau `array-contains-any` per kueri. Anda tidak dapat menggabungkan operator tersebut pada kueri yang sama.
- Anda tidak dapat menggabungkan `not-in` dengan tidak sama dengan `!=`.

- Anda tidak dapat mengurutkan kueri berdasarkan kolom yang disertakan dalam klausa kesetaraan (==) atau `in`.

15. MELAKUKAN QUERY GABUNGAN

Anda dapat merangkai beberapa metode operator kesetaraan (== atau `array-contains`) untuk membuat kueri yang lebih spesifik (`AND` yang logis). Namun, Anda harus membuat [indeks komposit](#) untuk menggabungkan operator kesetaraan dengan operator ketidaksetaraan, `<`, `<=`, `>`, dan `!=`.

```
citiesRef.where("state", "==", "CO").where("name", "==", "Denver");
citiesRef.where("state", "==", "CA").where("population", "<", 1000000)
;
```

Anda dapat melakukan perbandingan rentang (`<`, `<=`, `>`, `>=`) atau tidak sama dengan (`!=`) hanya di satu kolom, dan Anda dapat menyertakan maksimal satu klausa `array-contains` atau `array-contains-any` pada kueri gabungan:

Kueri grup koleksi

Grup koleksi terdiri dari semua koleksi dengan ID yang sama. Secara default, kueri mengambil hasil dari satu koleksi di database Anda. Gunakan kueri grup koleksi untuk mengambil dokumen dari grup koleksi, bukan dari satu koleksi.

Misalnya, Anda dapat membuat grup koleksi `landmarks` dengan menambahkan subkoleksi bangunan terkenal ke setiap kota:

```
var citiesRef = db.collection("cities");

var landmarks = Promise.all([
  citiesRef.doc("SF").collection("landmarks").doc().set({
    name: "Golden Gate Bridge",
    type: "bridge",
  }),
  citiesRef.doc("SF").collection("landmarks").doc().set({
    name: "Legion of Honor",
    type: "museum",
  }),
  citiesRef.doc("LA").collection("landmarks").doc().set({
    name: "Griffith Park",
    type: "park",
  }),
  citiesRef.doc("LA").collection("landmarks").doc().set({
    name: "The Getty",
    type: "museum",
  }),
  citiesRef.doc("DC").collection("landmarks").doc().set({
    name: "Lincoln Memorial",
    type: "memorial",
  }),
  citiesRef.doc("DC").collection("landmarks").doc().set({
    name: "National Air and Space Museum",
  })
]);
```

```

        type: "museum",
    })),
    citiesRef.doc("TOK").collection("landmarks").doc().set({
        name: "Ueno Park",
        type: "park",
    }),
    citiesRef.doc("TOK").collection("landmarks").doc().set({
        name: "National Museum of Nature and Science",
        type: "museum",
    }),
    citiesRef.doc("BJ").collection("landmarks").doc().set({
        name: "Jingshan Park",
        type: "park",
    }),
    citiesRef.doc("BJ").collection("landmarks").doc().set({
        name: "Beijing Ancient Observatory",
        type: "museum",
    }),
]);

```

Kita dapat menggunakan kueri sederhana dan gabungan yang dijelaskan sebelumnya untuk membuat kueri subkoleksi `landmarks` satu kota, tetapi Anda juga dapat mengambil hasil dari subkoleksi `landmarks` setiap kota sekaligus.

Grup koleksi `landmarks` terdiri dari semua koleksi dengan ID `landmarks`, dan Anda dapat membuat kuerinya menggunakan kueri grup koleksi. Misalnya, kueri grup koleksi ini mengambil semua bangunan terkenal `museum` di semua kota:

```

var museums = db.collectionGroup("landmarks").where("type", "==", "museum");
museums.get().then((querySnapshot) => {
    querySnapshot.forEach((doc) => {
        console.log(doc.id, " => ", doc.data());
    });
});

```

15. MELAKUKAN QUERY MEMBATASI DATA ORDER BY

Cloud Firestore menyediakan fungsionalitas kueri yang dapat diandalkan untuk menentukan dokumen mana yang ingin Anda ambil dari koleksi. Kueri ini juga dapat digunakan dengan `get()` atau `addSnapshotListener()`, seperti yang dijelaskan dalam [Mendapatkan Data](#).

```
//Misalnya, Anda dapat mengajukan kueri untuk 3 kota pertama menurut abjad de
ngan:
citiesRef.orderBy("name").limit(3);
//Anda juga dapat mengurutkan secara menurun untuk mendapatkan 3 kota terakhir:
citiesRef.orderBy("name", "desc").limit(3);
//Anda juga dapat mengurutkan berdasarkan beberapa kolom.Misalnya, jika Anda
ingin mengurutkan berdasarkan negara, dan dalam setiap negara mengurutkan ber
dasarkan populasi dalam urutan menurun:
citiesRef.orderBy("state").orderBy("population", "desc");
//Anda dapat menggabungkan filter where() dengan orderBy() dan limit().Dalam
contoh berikut, kueri menentukan ambang populasi, mengurutkan berdasarkan pop
ulasi dalam urutan menaik, dan hanya menampilkan beberapa hasil pertama yang
melebihi ambang batas:
citiesRef.where("population", ">", 100000).orderBy("population").limit(2);
```

Namun, jika Anda memiliki filter dengan perbandingan rentang (<, <=, >, >=), pengurutan pertama harus berada di kolom yang sama. Lihat daftar batasan `orderBy()` di bawah ini. Perhatikan batasan berikut untuk klausa `orderBy()`:

- Klausa `orderBy()` juga memfilter keberadaan kolom tertentu. Kumpulan hasil tidak akan menyertakan dokumen yang tidak berisi kolom yang dimaksud.
- Jika Anda menyertakan filter dengan perbandingan rentang (<, <=, >, >=), urutan pertama harus berada di kolom yang sama:

16. FILTER DATA TABLE, PAGINATION EXPORT IMPORT EXCEL

17 OPERASI FILE DALAM DATABASE

18. CANVAS HTML MENULIS DAN MENGGAMBAR

19. MAP DENGAN LEAFLET JS

20. GRAPHIC DENGAN CHARTJS