Universitas Semarang
Fakultas Teknologi Informasi dan Komunikasi
Teknik Informatika

Cross Platform System dengan Android dan Web Framework Flask

Modul Pelatihan SKPI Mahasiswa

Oleh:
Alauddin Maulana Hirzan, S. Kom., M. Kom
NIDN. 0607069401 | NIS. 06557003102238

Daftar Isi

Daftar Gambar

Pendahuluan

0.1 Android

Android merupakan sebuah sistem operasi yang terpasang di perangkat-perangkat mobile yang ditujukan untuk portabilitas dan mobilitas tinggi. Sehingga penggunanya dapat mengakses beberapa aplikasi maupun fitur di manapun dan kapanpun. Google sudah mengeluarkan beberapa versi dan jenis arsitektur Android untuk bisa menggapai berbagai macam platform mulai dari Komputer hingga Sistem Benam.

0.2 Kotlin

Kotlin adalah bahasa pemrograman yang digunakan oleh Android untuk membuat aplikasinya menggantikan bahasa pemrograman Java. Meski baru, bahasa ini mampu berinteraksi dengan script-script dari Java dengan baik.

0.3 Web Framework Flask

Flask adalah sebuah micro framework yang digunakan untuk mengatur penyajian Web dengan menggunakan teknologi *Web Server Gateway Interface* (WSGI) tanpa menggunakan *tools* atau *libraries* khusus layaknya framework pada umumnya. Sehingga dalam penggunaanya tidak memerlukan database tersendiri seperti XAMPP maupun WAMPP.

0.4 Firebase Cloud Firestore

Cloud Firestore merupakan database berjenis *NoSQL* yang disediakan oleh Google yang bisa diakses secara gratis oleh user. Database jenis ini mempu menyimpan data dalam bentuk *Dokumen dan Koleksi*, sehingga cocok untuk dijadikan tempat menyimpan data yang kompleks. Untuk mengakses database ini, dapat dilakukan via web console Google maupun melalui pemrograman

Persiapan Pelatihan

Agar pelatihan dapat berjalan dengan lancar, mahasiswa diwajibkan memenuhi persyaratan berikut baik dalam bentuk perangkat keras maupun lunak:

0.5 Perangkat Keras

Android Studio merupakan IDE yang membutuhkan resource yang cukup besar sehingga untuk membuat aplikasi mobile ini membutuhkan spesifikasi yang harus mencukupi.

- Prosesor 4 Inti
- Memori RAM 4GB
- Harddisk 500GB
- Keyboard
- Mouse

0.6 Perangkat Lunak

Perangkat lunak berikut ini wajib diinstall oleh mahasiswa demi lancarnya praktikum:

- Android Studio Terbaru https://developer.android.com/studio
- Web Browser
- ADB Driver Windows https://adb.clockworkmod.com/

0.7 File Tambahan

Unduh file-file berikut untuk membantu proses pembuatan WebApp

- https://s.id/CPA-SKPI

Bab 1

Mobile Application

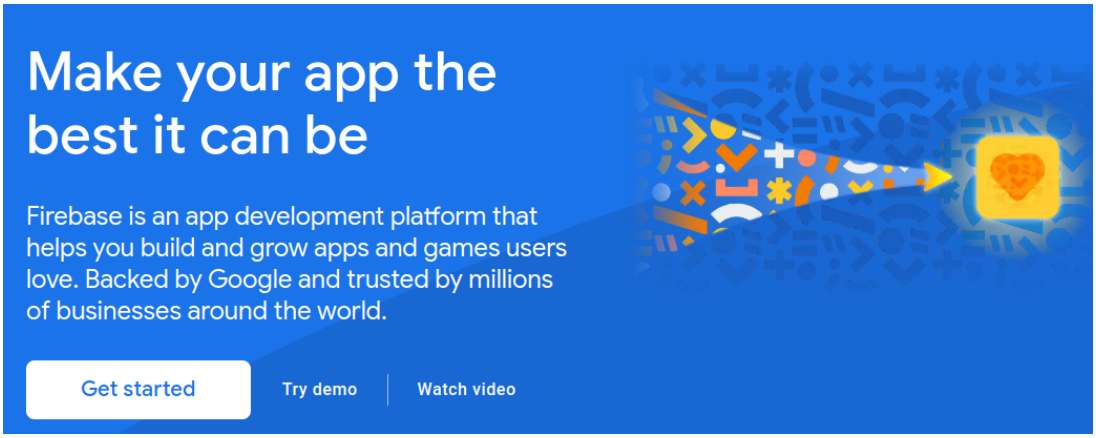
1.1 Android + Cloud Firestore

Di bagian ini mahasiswa diajarkan bagaimana membuat projek Google yang digunakan untuk membuat database dengan Cloud Firestore. Mahasiswa diwajibkan memiliki Akun Google untuk bisa membuat ini

1.2 Tutorial

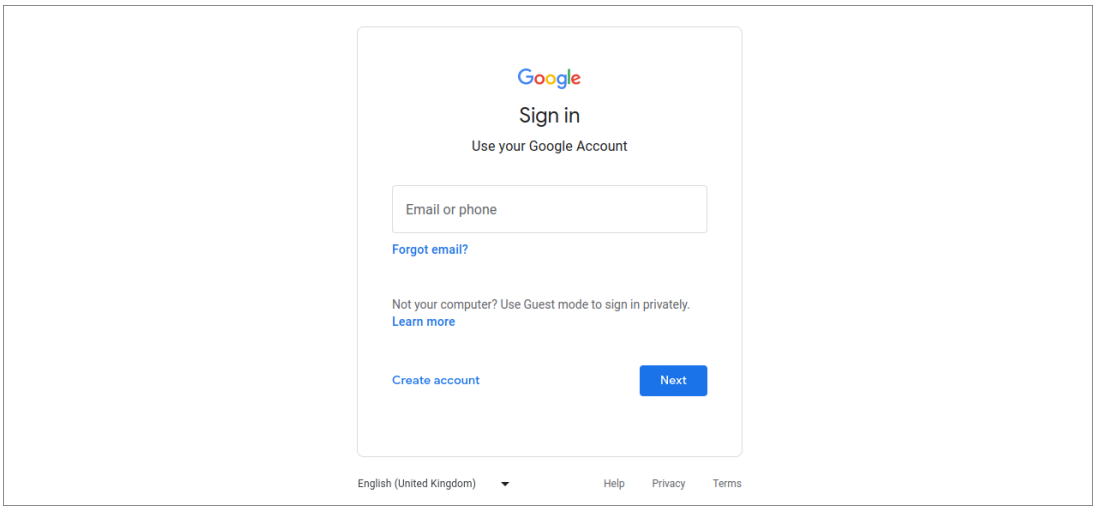
1.2.1 Membuat Database Cloud Firestore

1. Dengan menggunakan **Browser**, buka website Firebase melalui link : https://firebase.google.com/
2. Lalu klik ***Get started***



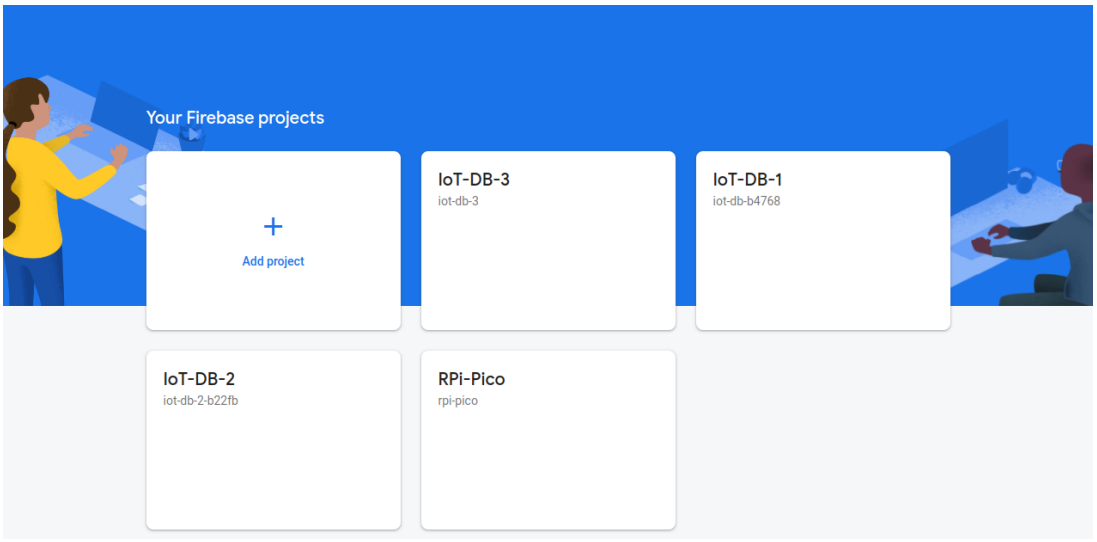
Gambar 1.1: Halaman Firebase

3. Login dengan menggunakan Akun Google pribadi masing-masing



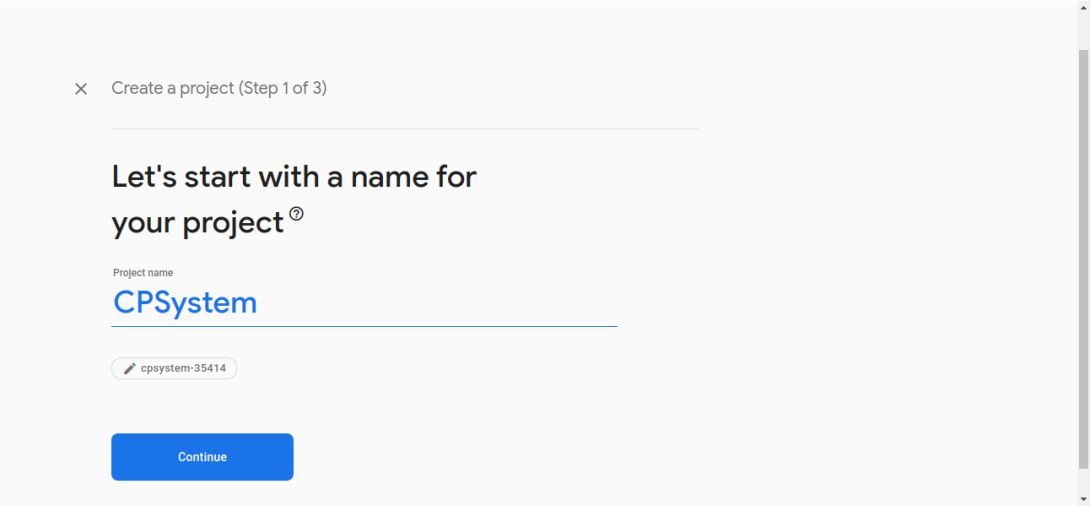
Gambar 1.2: Halaman Login Google

4. Google kemudian akan menampilkan projek-projek yang digunakan untuk Firebase. Klik ***Add project***



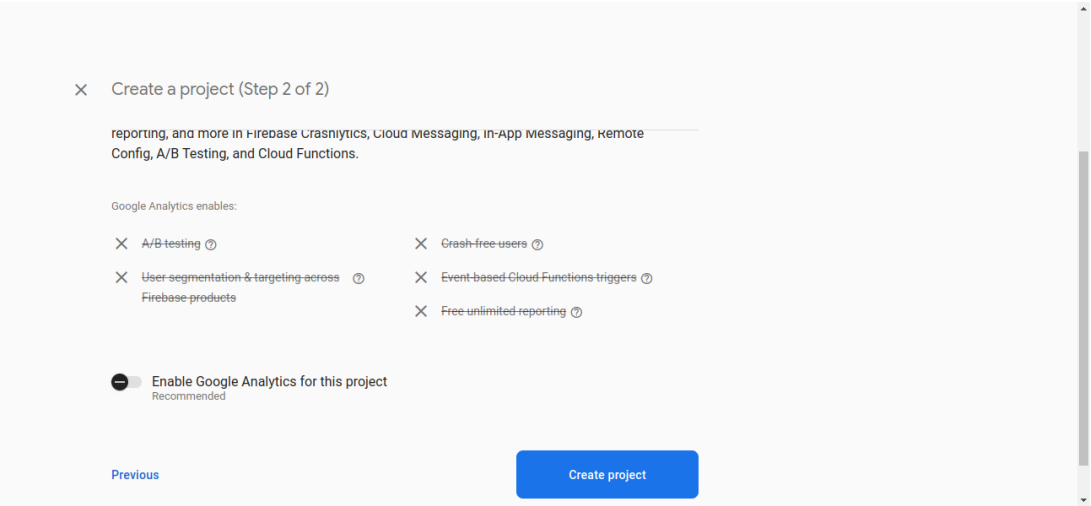
Gambar 1.3: Halaman Penambahan Projek

5. Firebase akan mulai membuatkan projeknya. Masukkan nama projek di kolom yang disediakan. Lalu klik ***Continue***



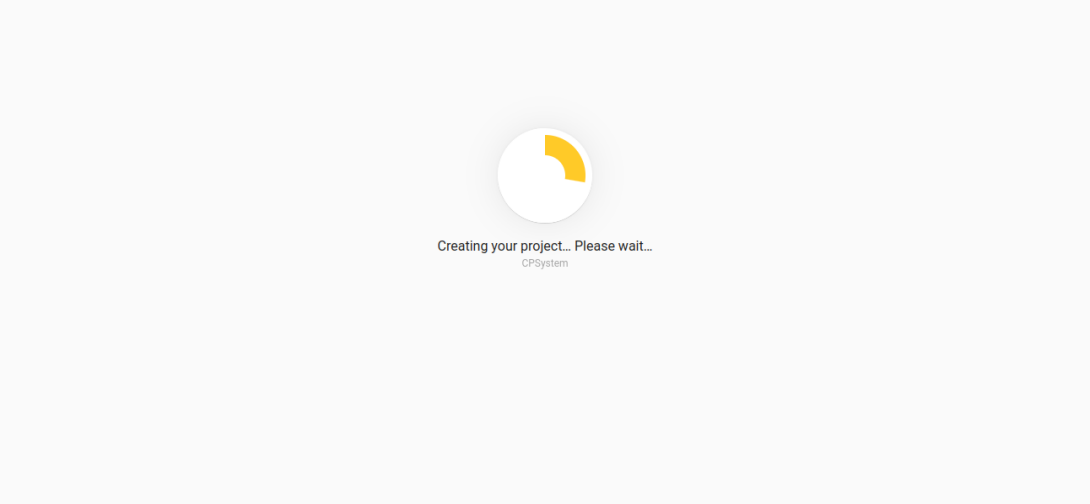
Gambar 1.4: Halaman Nama Projek

6. Bagian berikutnya adalah *Google Analytic* dan tidak diperlukan di projek ini. **Matikan *Analytic*** lalu klik ***Create Project***



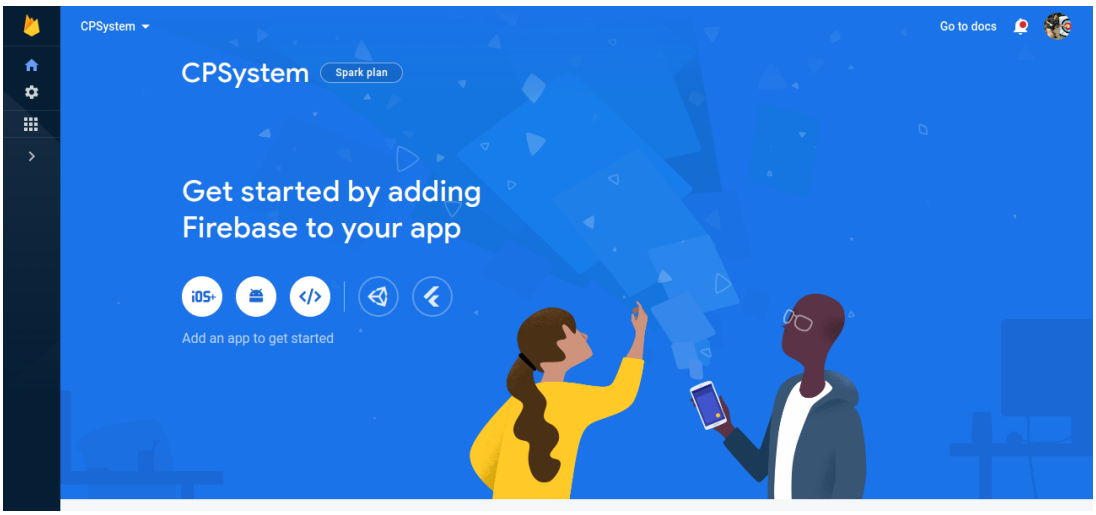
Gambar 1.5: Halaman Pembuatan Projek

7. Tunggu Firebase membuat projek. Klik ***Continue*** jika sudah selesai



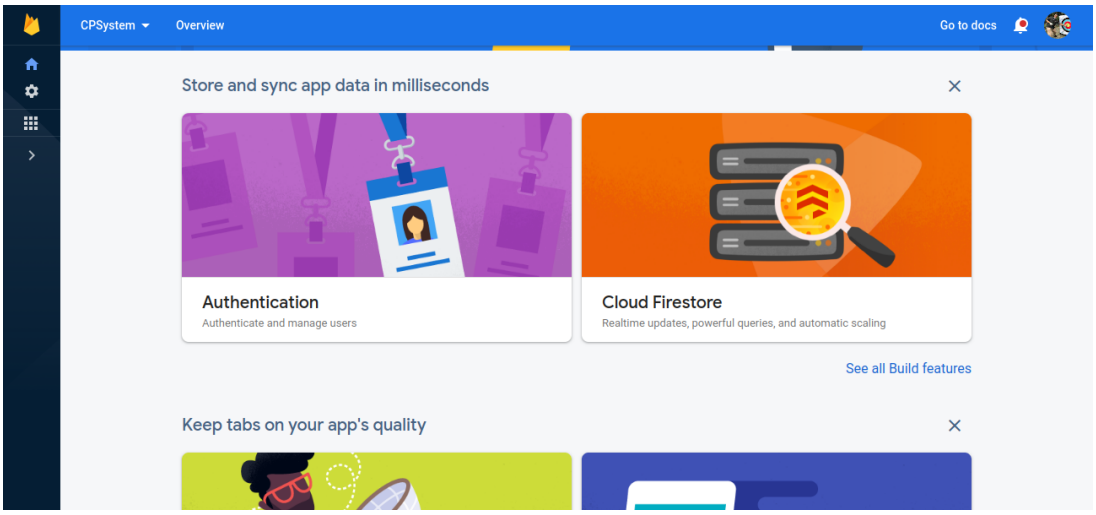
Gambar 1.6: Halaman Pemrosesan Projek

8. Firebase akan menampilkan projek yang sudah selesai



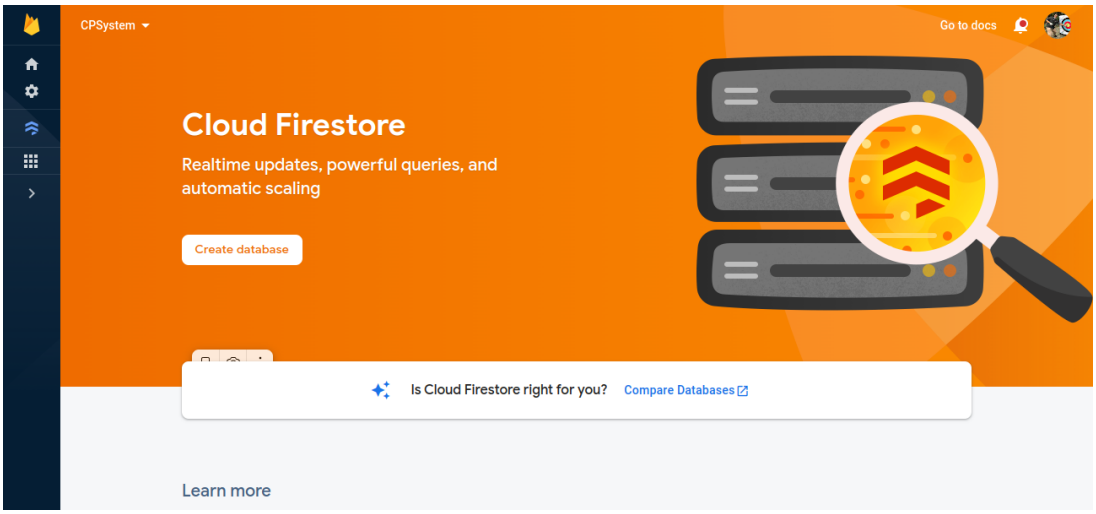
Gambar 1.7: Halaman Dashboard Projek

9. Turunkan halaman dan cari ***Cloud Firestore***. **Klik** aplikasi tersebut untuk membuka database.



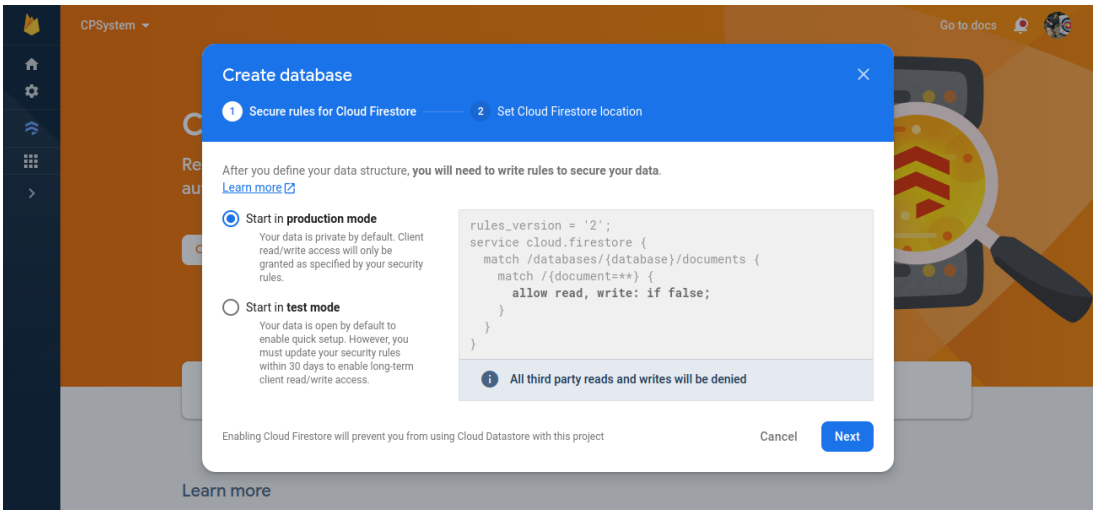
Gambar 1.8: Halaman Pemilihan Aplikasi

10. Setelah diklik, Firebase akan membuka halaman dasbor ***Cloud Firestore***. **Klik *Create database*** untuk membuat database baru



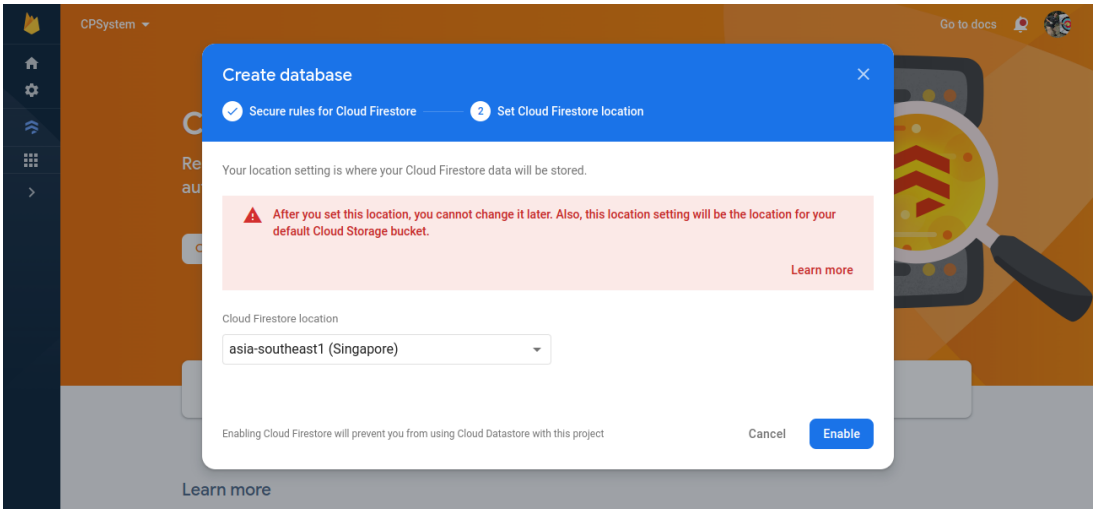
Gambar 1.9: Halaman Dashboard Firestore

11. Untuk membuat database, mahasiswa diminta memilih mode keamanan. **Pilih *Start in production mode***. Lalu **Klik *Next***



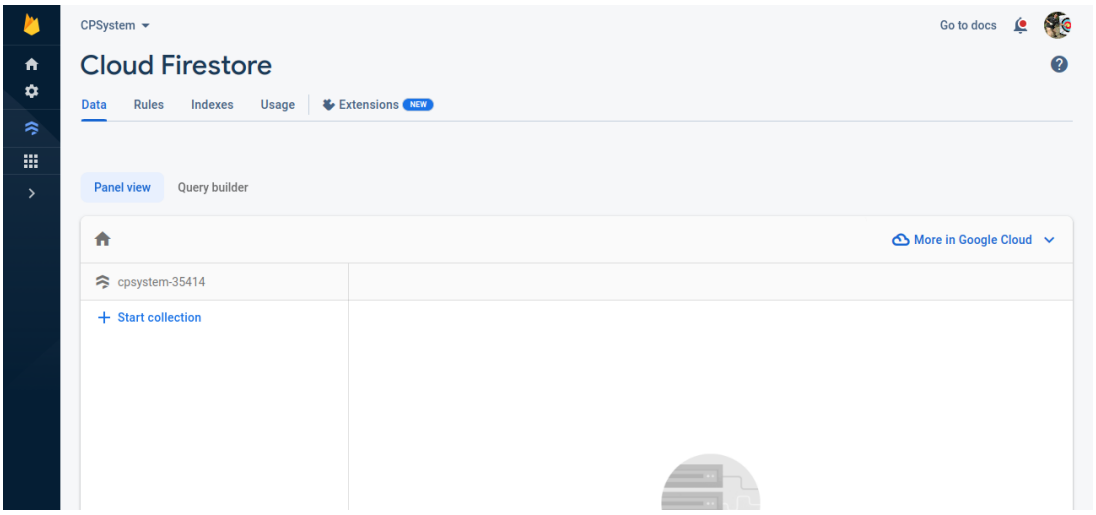
Gambar 1.10: Halaman Pemilihan Mode Keamanan

12. Langkah berikutnya adalah memilih lokasi penyimpanan. **Pilih *asia-southeast1 (Singapore)***. Lalu **Klik *Enable***



Gambar 1.11: Halaman Pemilihan Lokasi Server

13. Tunggu Firebase melakukan pembuatan Database Firestore.
14. ***Cloud Firestore*** yang sudah siap digunakan akan menampilkan halaman sebagai berikut

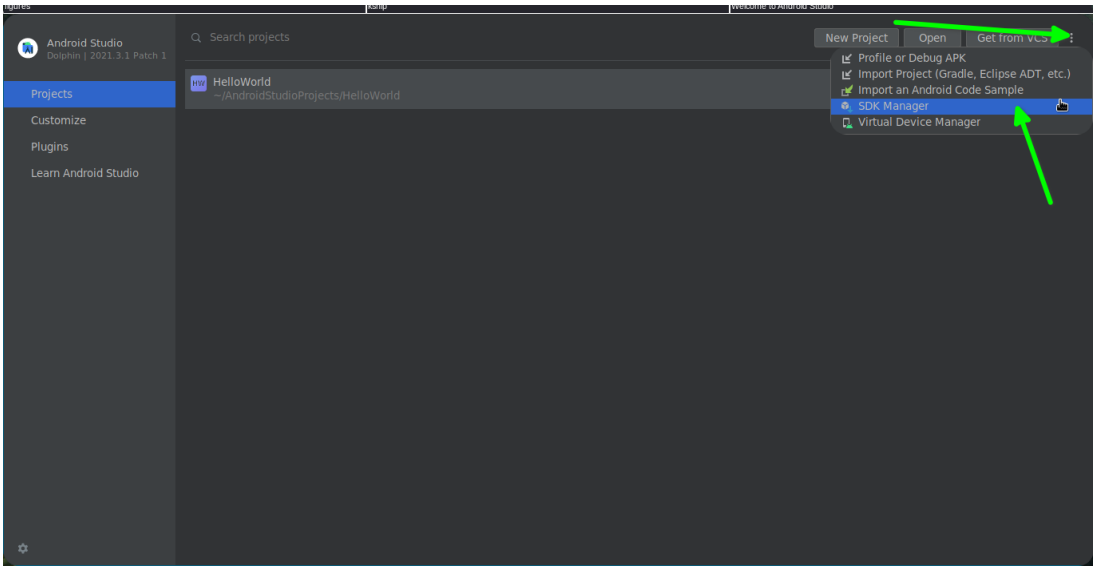


Gambar 1.12: Halaman Database Firestore

15. Di sinilah nanti data akan disimpan menurut jenisnya. Data bisa dipisah menurut jenisnya.

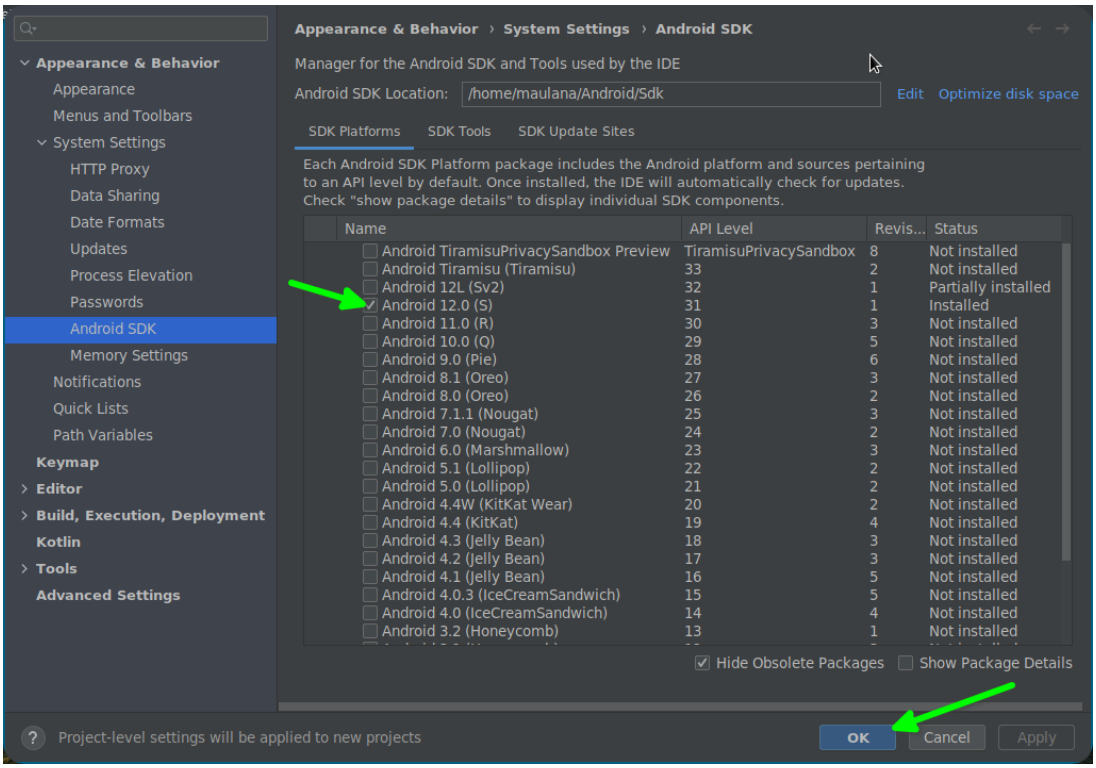
1.2.2 Aplikasi Android #1 - Penambahan Inventori

1. Untuk memulai membuat aplikasi Android, bukalah **Android Studio**. (Demi kelancaran, mohon menggunakan Android Studio terbaru). Untuk memastikan **SDK** sudah terinstall, klik **Tiga Titik** di bagian atas kanan. Lalu Klik **SDK Manager**



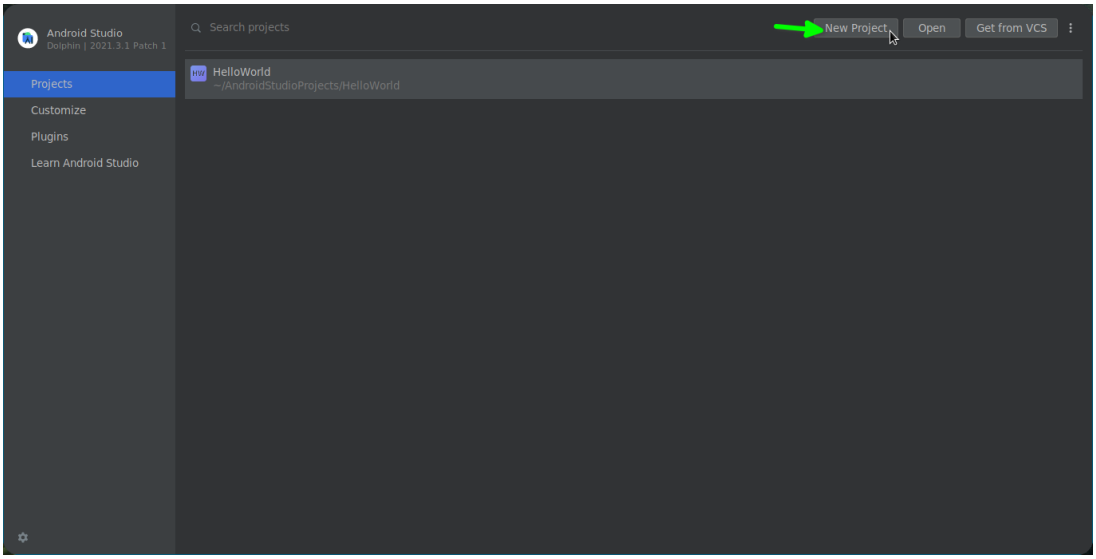
Gambar 1.13: Android : Android Studio

2. Pastikan setidaknya **Satu (1)** SDK yang terpasans .id / CPA-SKPI g di sistem. Jika **Tidak** install salah satu SDK.



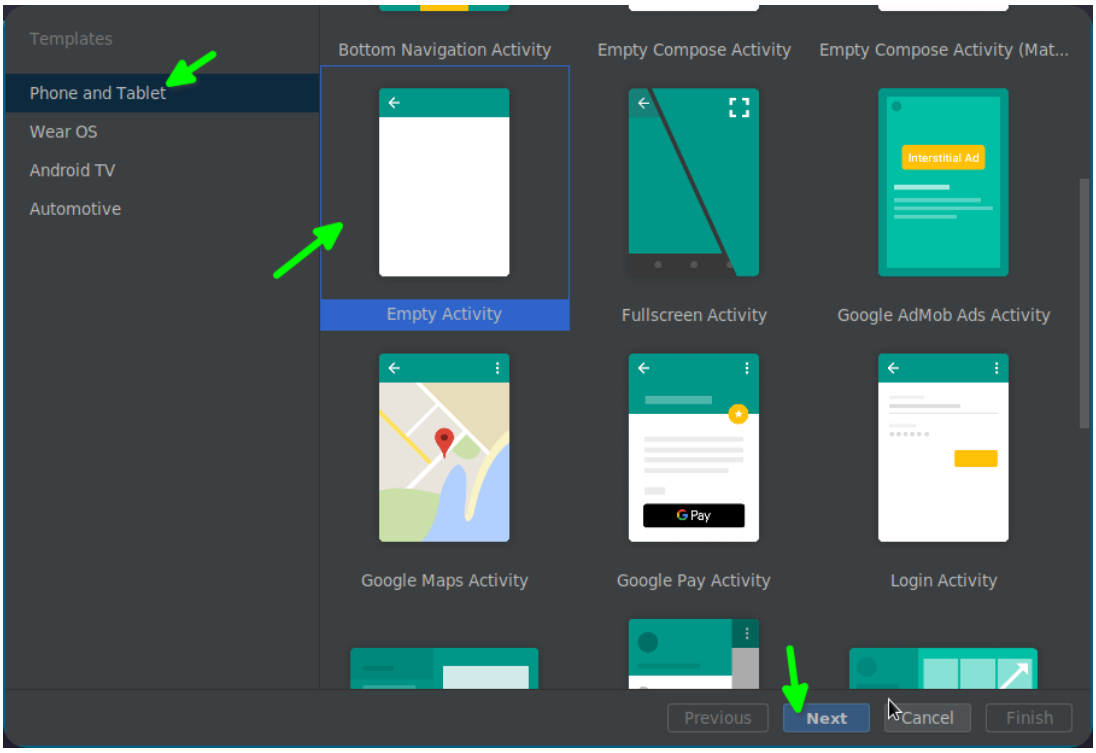
Gambar 1.14: Android : Instalasi SDK

3. Jika sudah siap, kembali ke tampilan awal Android Studio (Langkah 1), dan Klik **New Project**



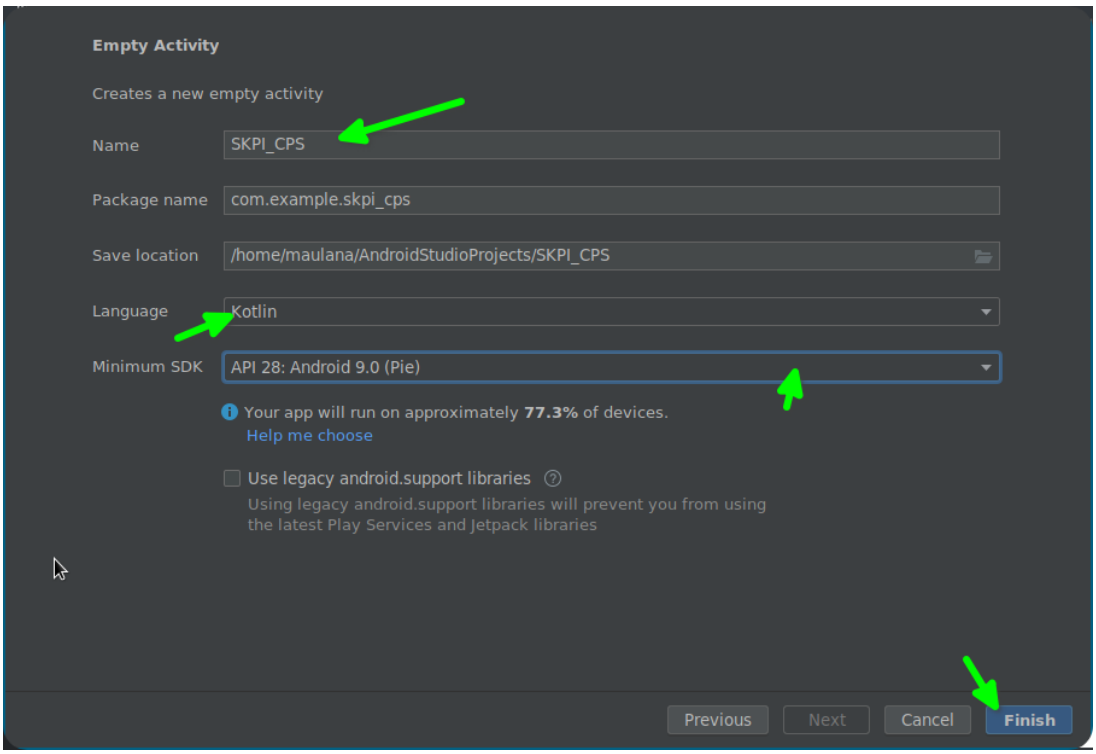
Gambar 1.15: Android : Projek baru

4. Di bagian pemilihan layout, pilih **Phone and Table**, dan **Empty Activity**, Klik **Next**



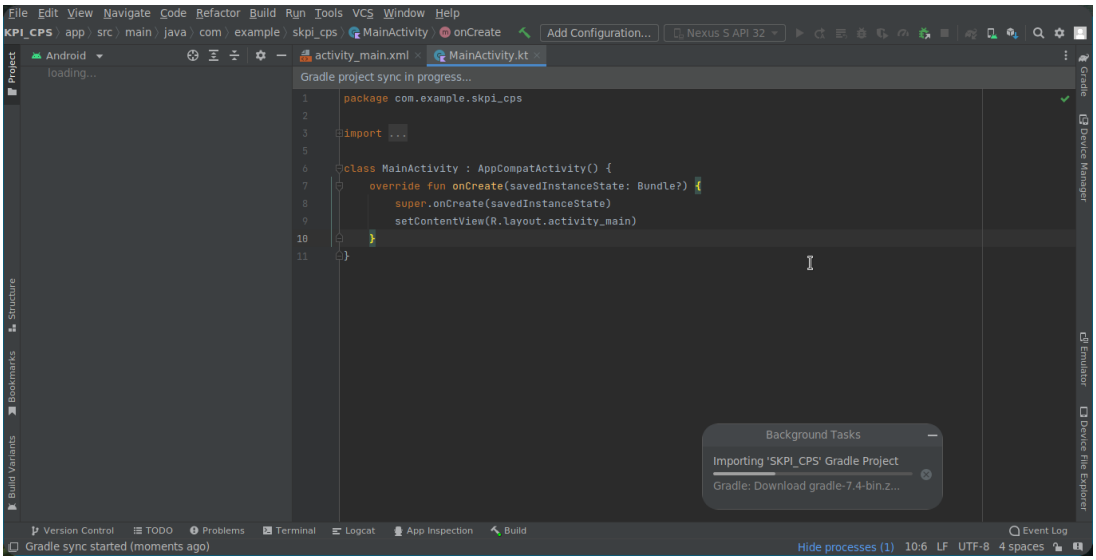
Gambar 1.16: Android : Memilih Perangkat dan Template

5. Ganti nama projek sesuai pilihan pribadi. Namun pastikan **Bahasa : Kotlin**, **SDK : Sesuai perangkat masing-masing**. Klik **Finish**



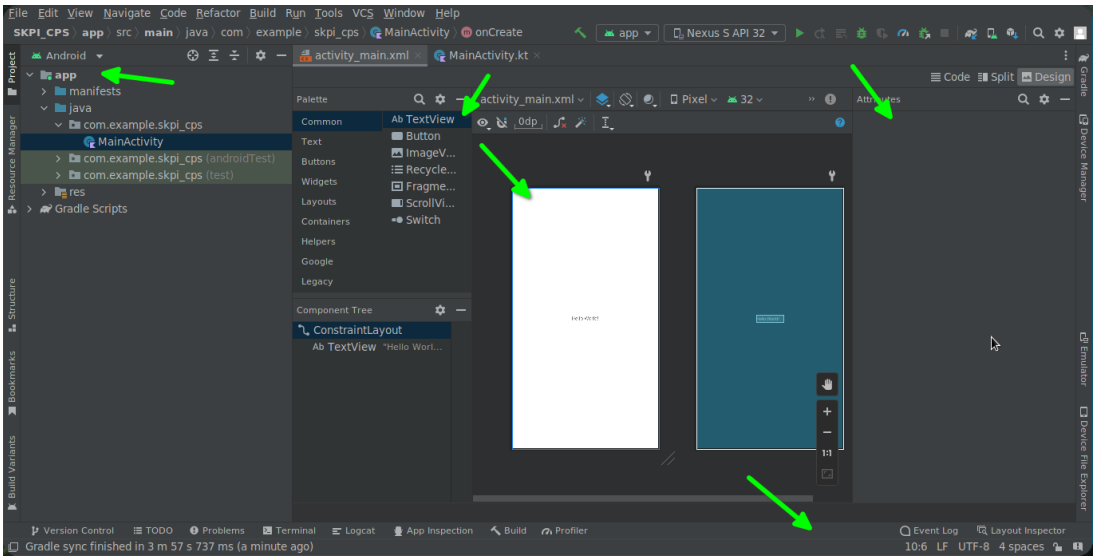
Gambar 1.17: Android : Konfigurasi Projek Android

6. Android Studio akan mulai membuat projek. Proses ini akan membutuhkan koneksi dan proses yang cukup lama. **Tunggu Sampai Selesai**



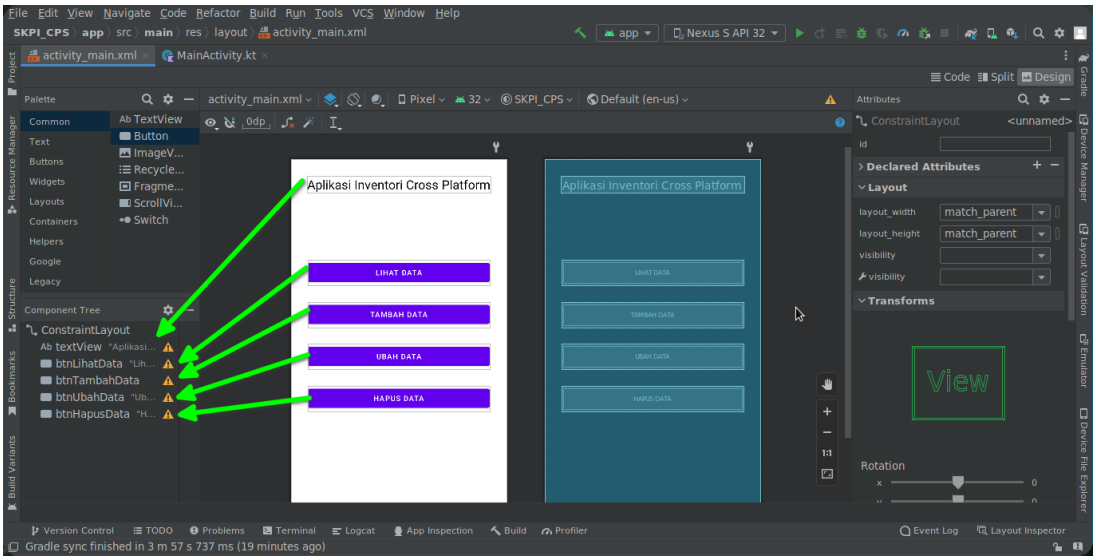
Gambar 1.18: Android : MainActivity Android

7. Android Studio yang sudah selesai melakukan sinkronisasi, akan menampilkan keseluruhan elemennya dengan benar.



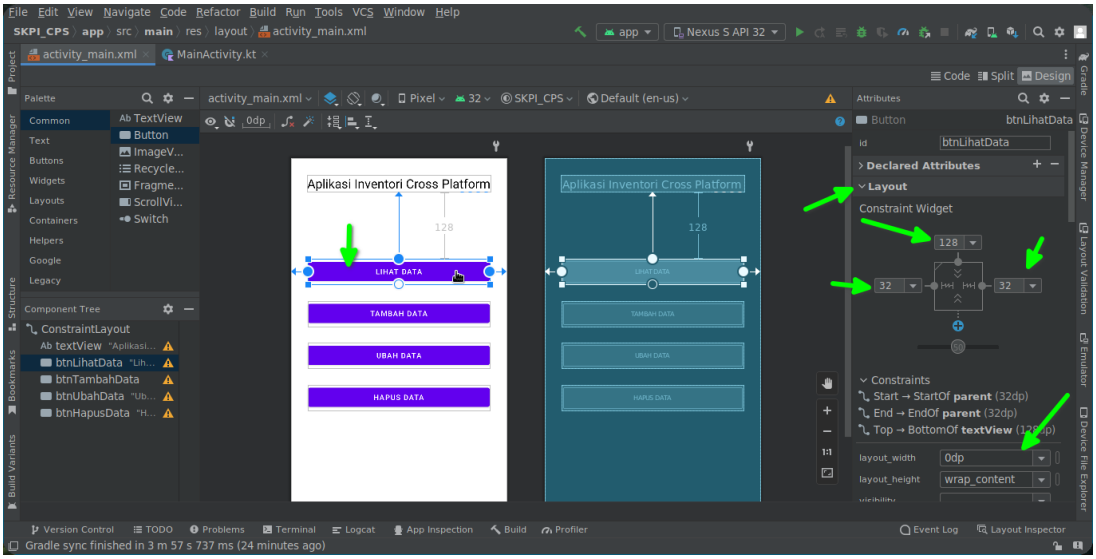
Gambar 1.19: Android : Layout MainActivity

8. Berikutnya adalah membuat tampilan terlebih dahulu. Gunakan **Drag'n Drop** untuk mengambil objek tampilan ke Layout Aplikasi. Ubah **id** dan **text** sesuai contoh di gambar. Berikut hasil akhirnya:



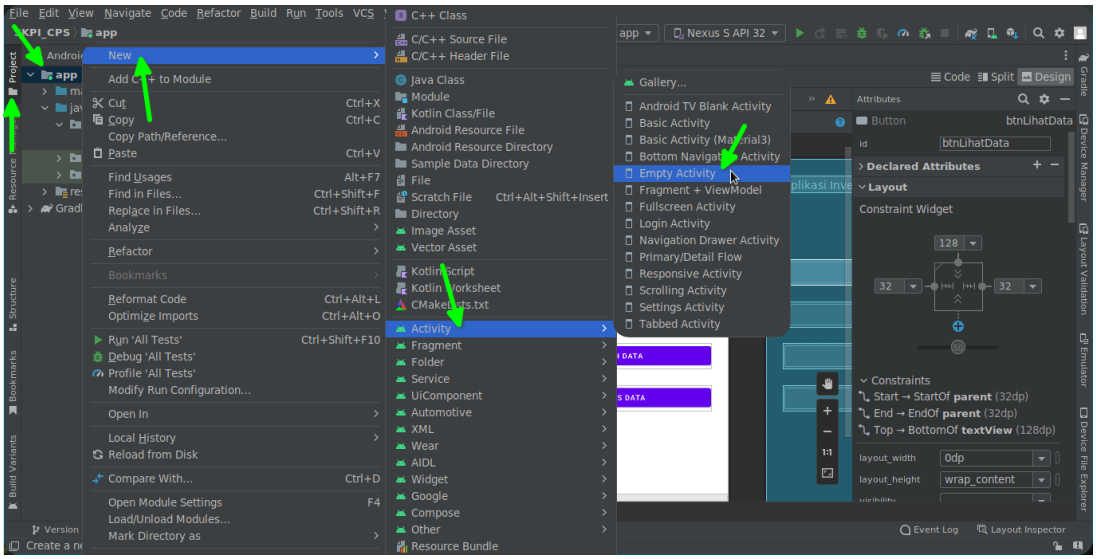
Gambar 1.20: Android : Layout MainActivity dengan Button

9. Untuk mengkonfigurasikan **Constraint** masing-masing objek, gunakan panel **Attribute** di sebelah kanan



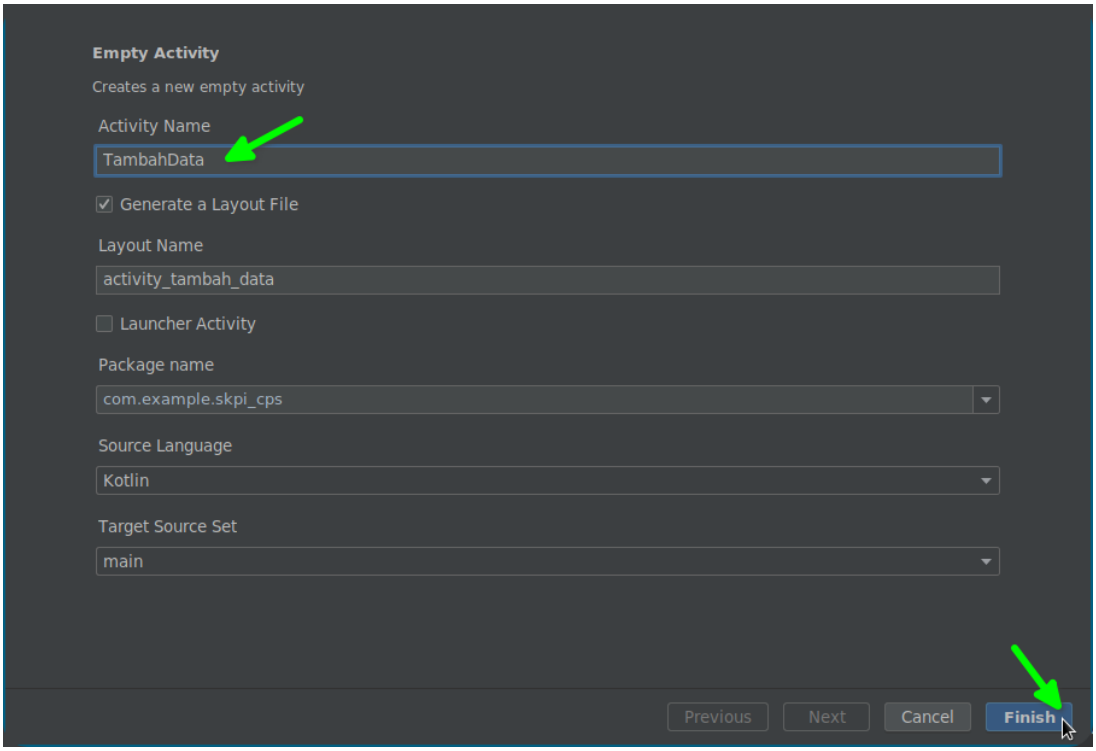
Gambar 1.21: Android : Pengaturan Constraint

10. Berikutnya adalah membuat **Activity** baru untuk masing-masing Tombol. Buka panel **Project**. Klik Kanan **App**. Pilih **New** => **Activity** => **Empty Activity**



Gambar 1.22: Android : Penambahan Activity Baru

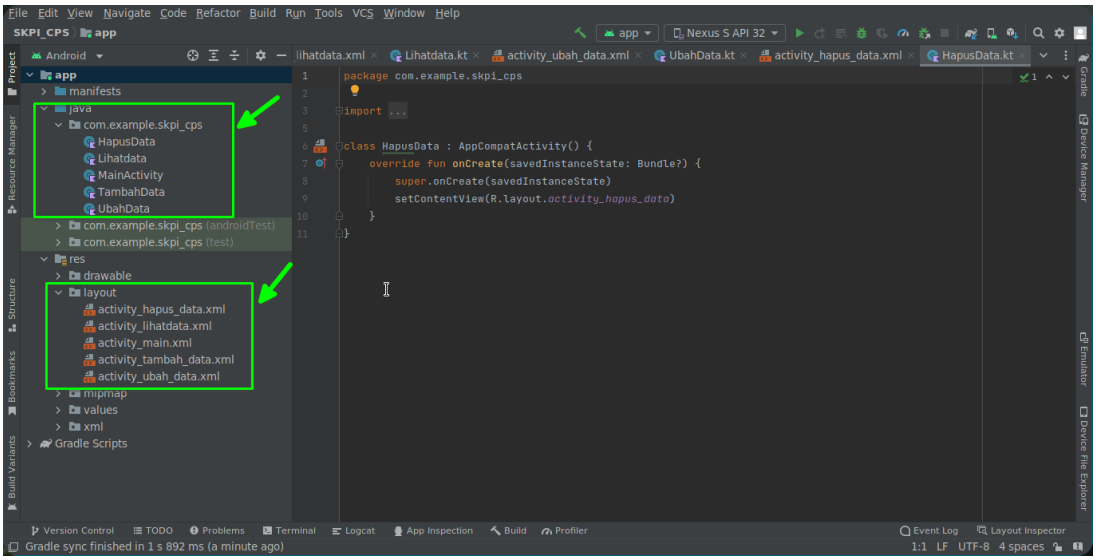
11. Ubah nama Activity menjadi **TambahData**. Lalu klik **Finish**



Gambar 1.23: Android : Konfigurasi Activity Baru

12. **Ulangi** untuk setiap fungsi **Tombol**
    - LihatData
    - TambahData
    - UbahData
    - HapusData

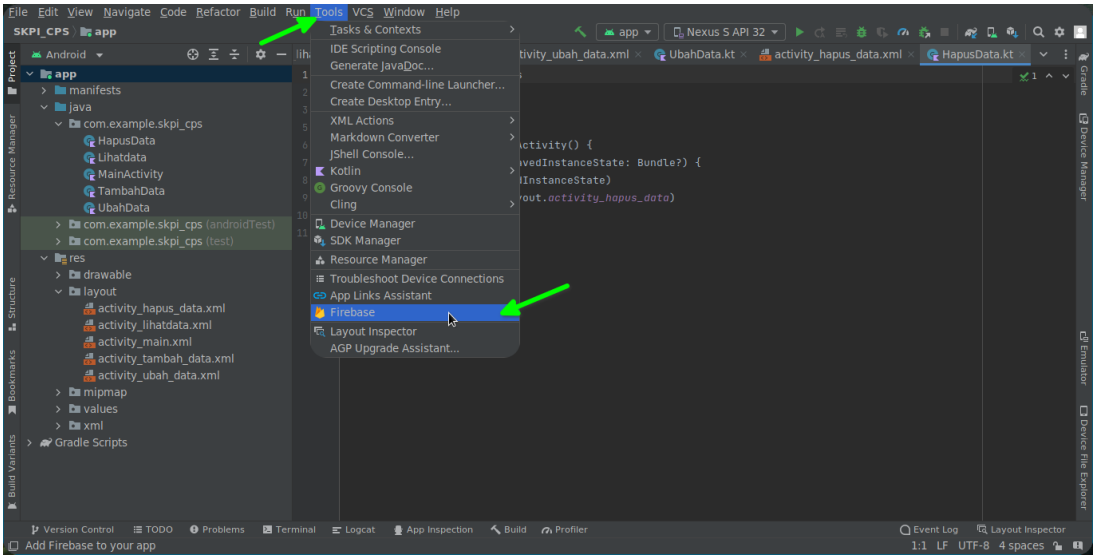
13. Hasil akhir penambahan Activity baru



Gambar 1.24: Android : Hasil Akhir Activity Baru

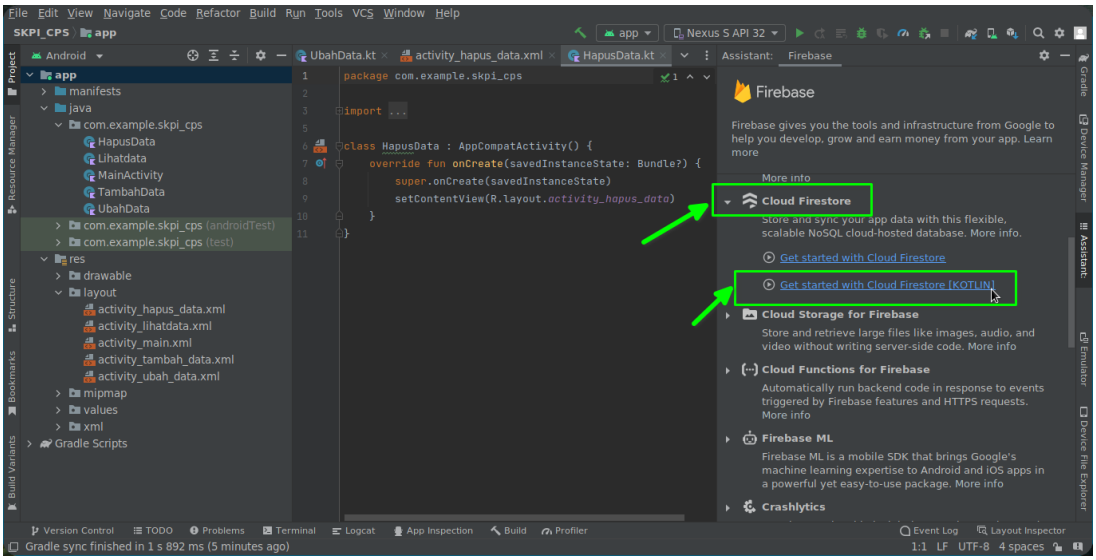
14. Sebelum melanjutkan ke tahap berikutnya. Sebaiknya menghubungkan **Firebase** ke **Android Studio**. Peringatan: Sambungan **Akun** ini berlaku per Projek. Bergantian memakai komputer dapat menyebabkan kebingungan database

15. Untuk menghubungkan akun **Firebase**, klik **Tools** => **Firebase**



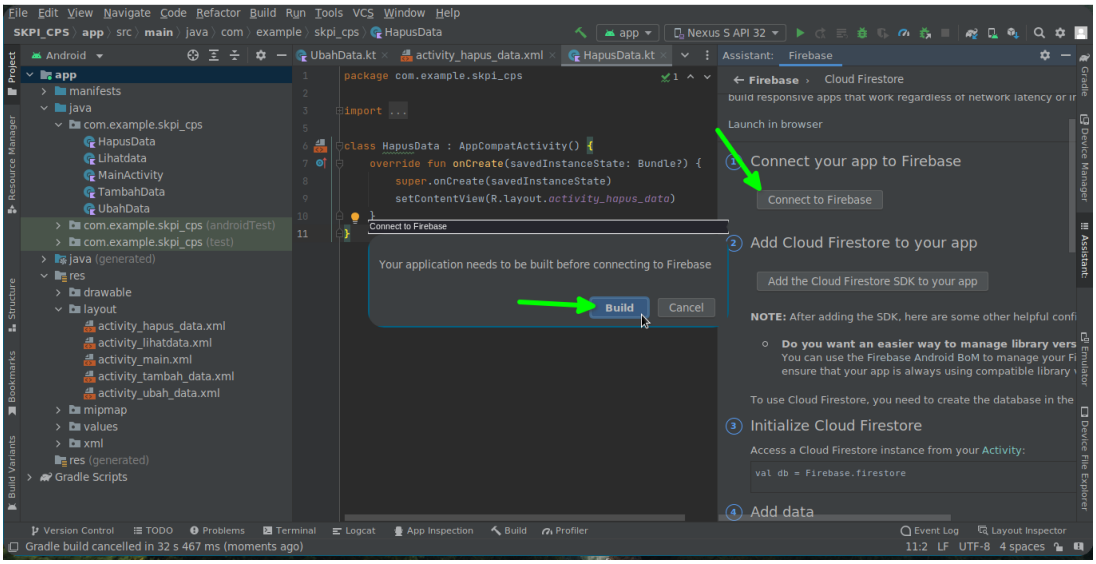
Gambar 1.25: Android : Penghubung Firebase

16. Cari **Cloud Firestore**, dan klik **Get started with Cloud Firestore (KOTLIN)**



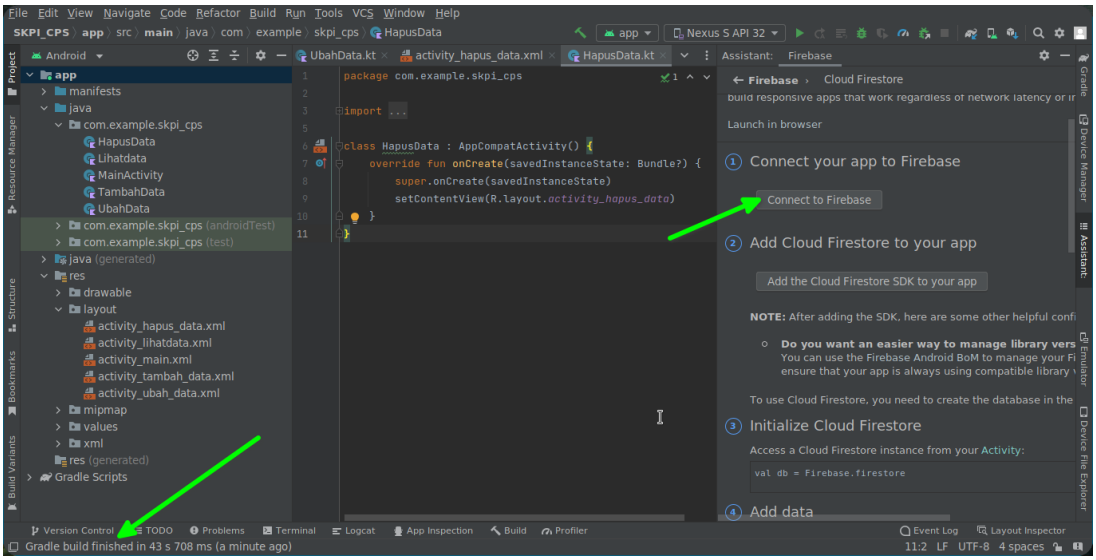
Gambar 1.26: Android : Menghubungkan Firebase Kotlin

17. Klik **Connect to Firebase**, dan akan muncul peringatan **Build**. **Klik Build**



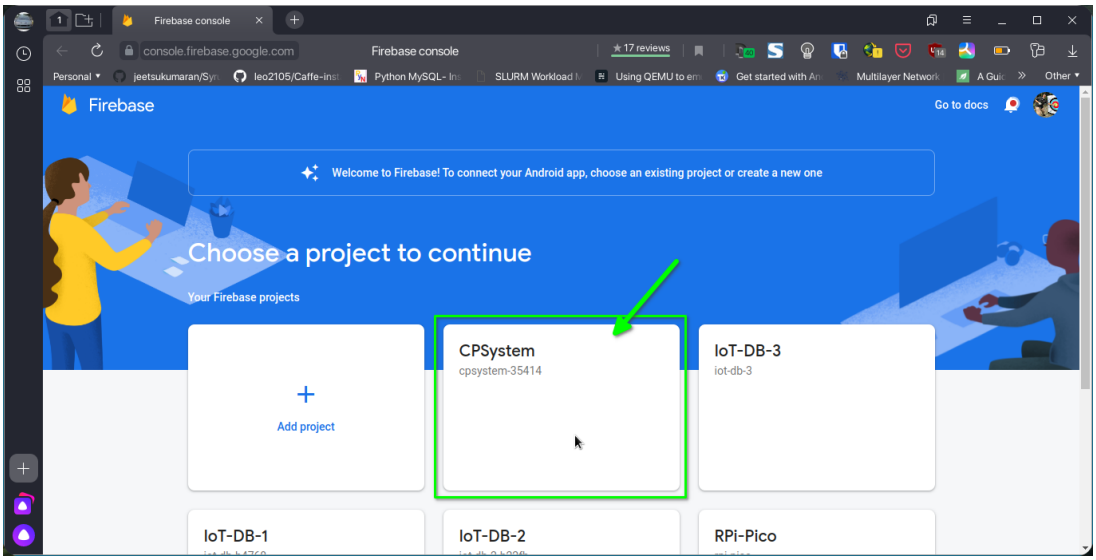
Gambar 1.27: Android : Peringatan Build Firebase

18. Tunggu hingga proses **Building** selesai. Jika sudah muncul **Gradle build finished** di pojok kiri bawah, maka projek siap dihubungkan. Klik **Connect to Firebase** lagi.



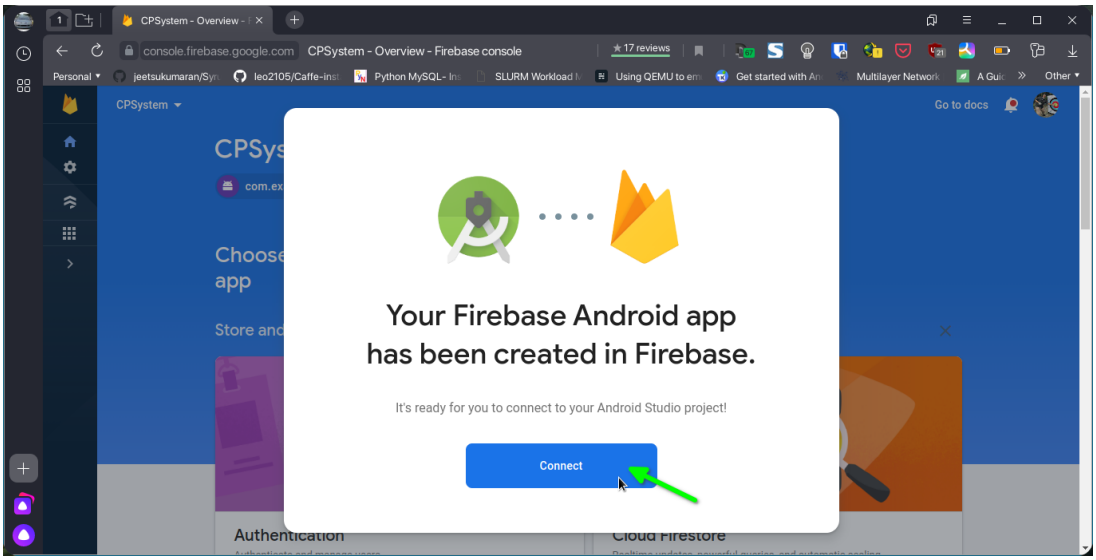
Gambar 1.28: Android : Menghubungkan Firebase Ulang

19. Android Studio akan membuka **Cloud Console Firebase** di **Browser Default Windows** untuk memilih projek Firebase dil angkah sebelumnya.



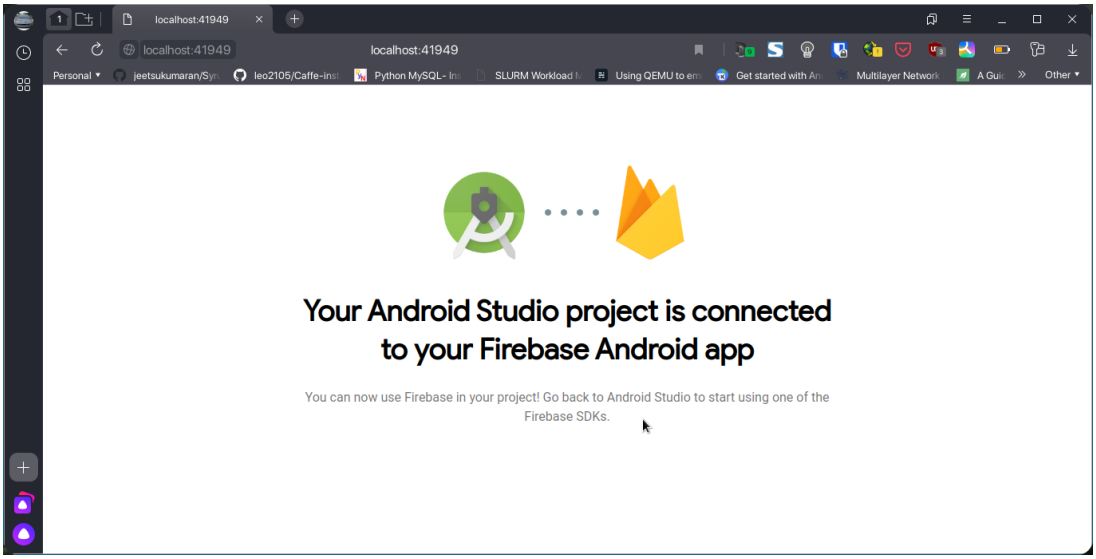
Gambar 1.29: Android : Memilih Projek Firebase

20. Klik **Connect** untuk menghubungkan



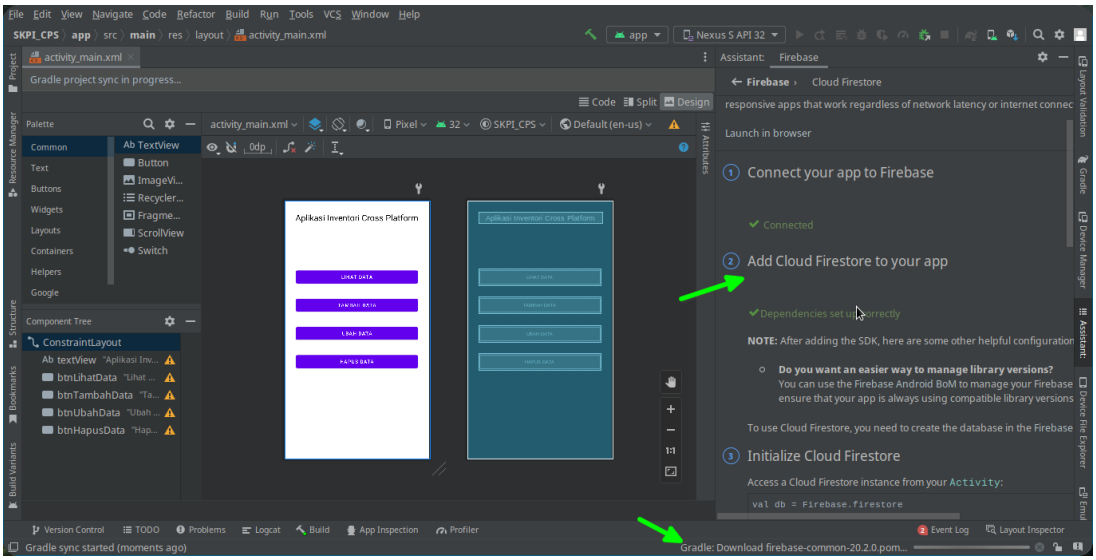
Gambar 1.30: Android : Menghubungkan Firebase ke Android Studio

21. Ketika sukses, kembali ke Android Studio



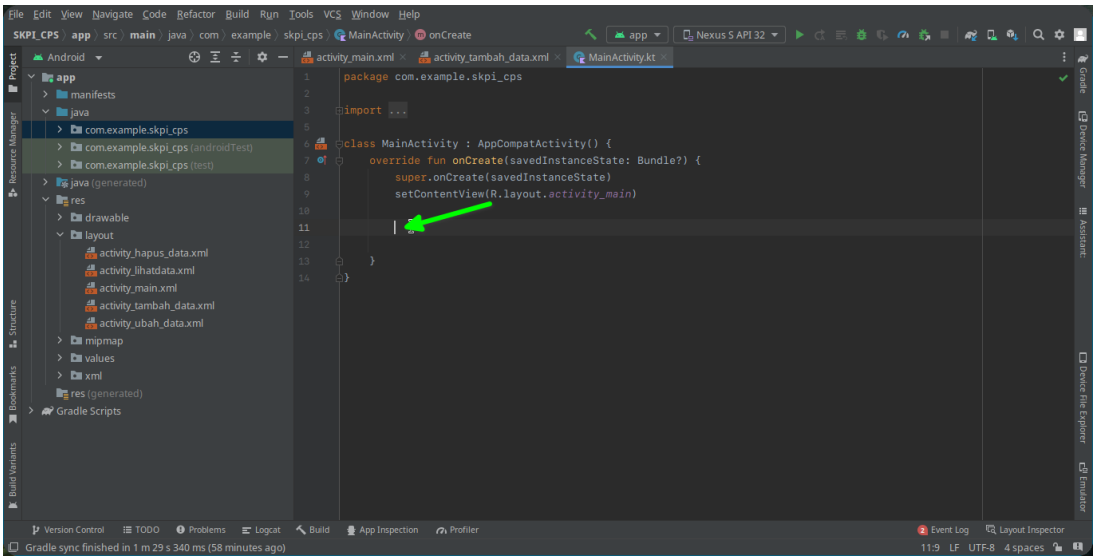
Gambar 1.31: Android : Firebase Terhubung

22. Berikutnya adalah menambahkan **SDK Firebase** ke Android Studio. Bisa dilakukan dengan cara mudah dengan klik **Add the Cloud Firestore SDK to your app** => Klik **Accept Changes**. Android Studio akan mendownload SDK



Gambar 1.32: Android : Menambahkan Dependency

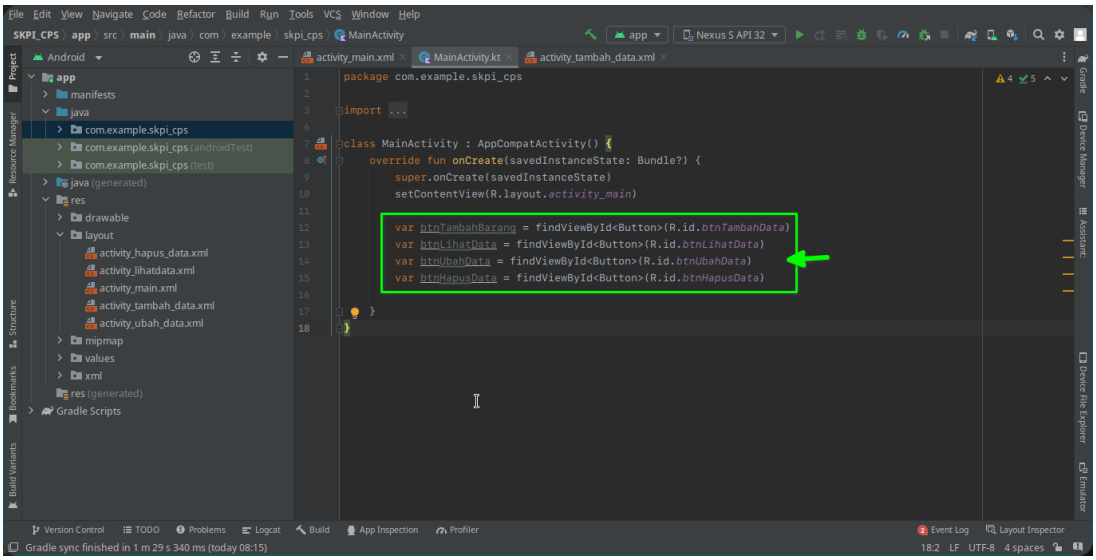
23. Jika sudah selesai dan semua sudah tercentang, buka file **MainActivity.kt**, tambahkan kode berikut ini tepat di tempat yang ditunjukkan panah



Gambar 1.33: Android : Lokasi Kode MainActivity

Potongan Kode

```
var btnTambahBarang = findViewById<Button>(R.id.btnTambahData)
var btnLihatData = findViewById<Button>(R.id.btnLihatData)
var btnUbahData = findViewById<Button>(R.id.btnUbahData)
var btnHapusData = findViewById<Button>(R.id.btnHapusData)
```

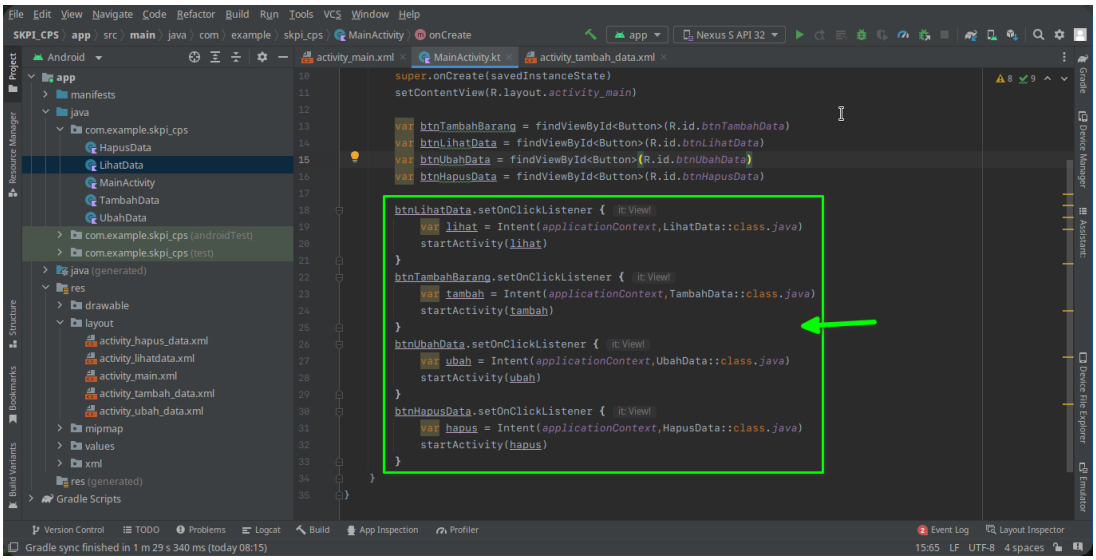


Gambar 1.34: Android : Hasil Kode Inisialisasi

24. Di bagian bawah kode tersebut, tambahkan **Fungsi Listener** untuk bisa membuka **Activity Baru**

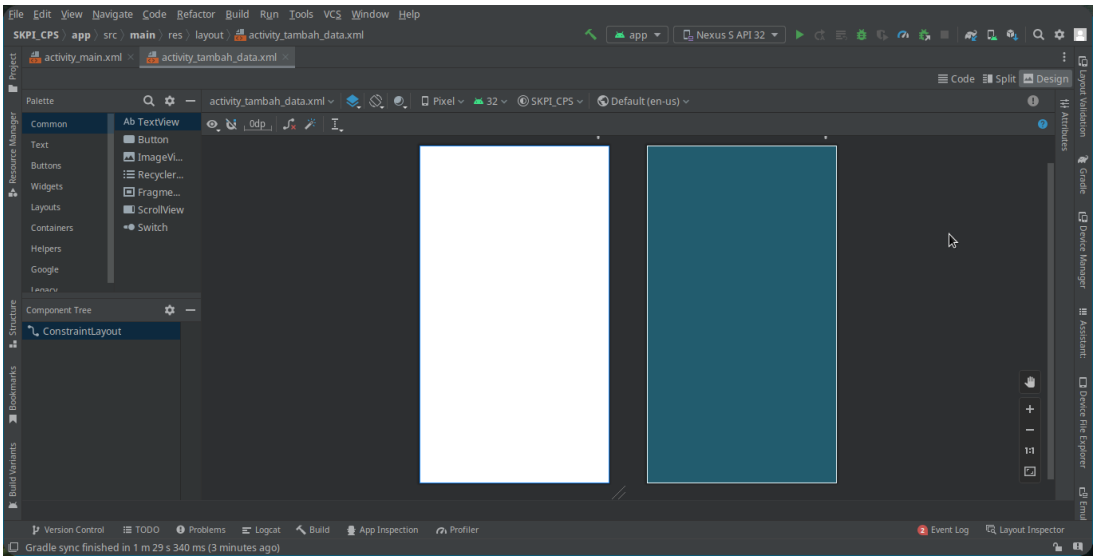
Potongan Kode

```
btnLihatData.setOnClickListener {
        var lihat = Intent(applicationContext,LihatData::class.java)
        startActivity(lihat)
}
btnTambahBarang.setOnClickListener {
        var tambah = Intent(applicationContext,TambahData::class.java)
        startActivity(tambah)
}
btnUbahData.setOnClickListener {
        var ubah = Intent(applicationContext,UbahData::class.java)
        startActivity(ubah)
}
btnHapusData.setOnClickListener {
        var hapus = Intent(applicationContext,HapusData::class.java)
        startActivity(hapus)
}
```



Gambar 1.35: Android : Hasil Kode Button

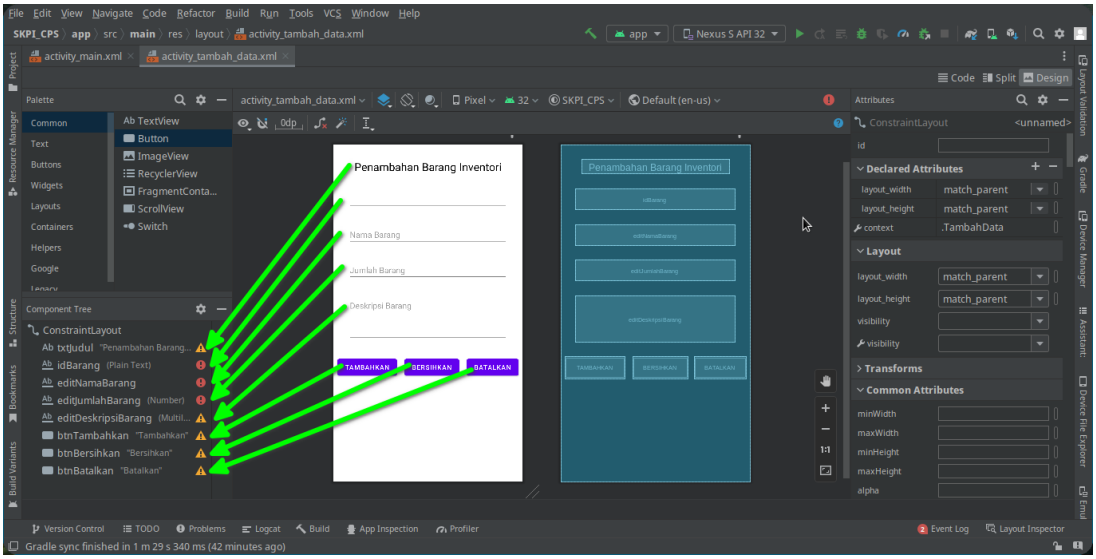
25. Berikutnya membuat layout TambahData. Buka file **activity_tambah_data.xml** (Bisa melalui **app** => **res** => **layout**)



Gambar 1.36: Android : Membuat Layout TambahData

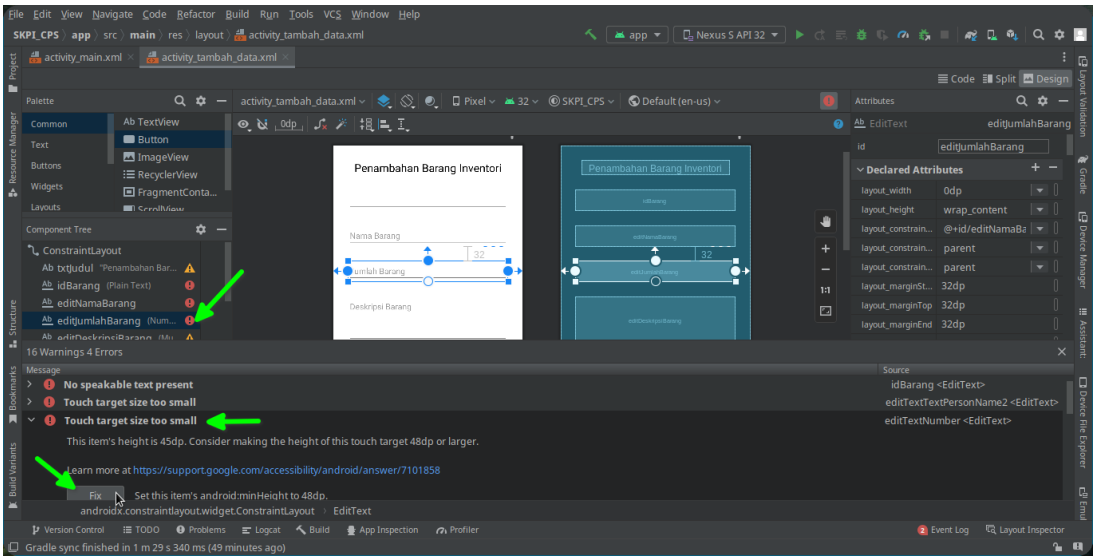
26. Buatlah tampilan dengan komposisi:
    - txtJudul - **TextView** (Judul)
    - idBarang - **Plain Text** (Attributes: **editable false**)
    - editNamaBarang - **Plain Text** (Attributes: **hint** bukan **text**)
    - editJumlahBarang - **Number** (Attributes: **hint** bukan **text**)
    - editDeskripsiBarang - **Multiline Text** (Attributes: **hint** bukan **text**)
    - btnTambahkan - **Button**
    - btnBersihkan - **Button**

- btnBatalkan - **Button**



Gambar 1.37: Android : Hasil Akhir TambahData

27. Jika muncul error seperti gambar di atas, cukup klik lingkaran merah tersebut dan betulkan error yang ada. Pilih **Fix** untuk masalah **Small Size** dan **Ignore** untuk **Speakable** error

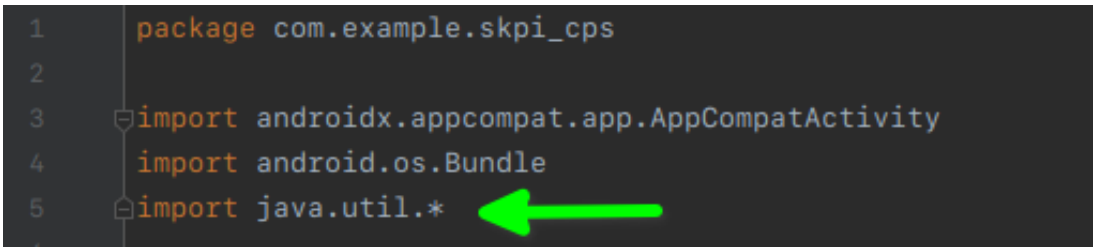


Gambar 1.38: Android : Perbaikan Error

28. Berikutnya adalah memrogram **Activity TambahData** tersebut. Buka file **TambahData.kt** lalu masukkan kode berikut:

Potongan Kode

```
import java.util.*
```

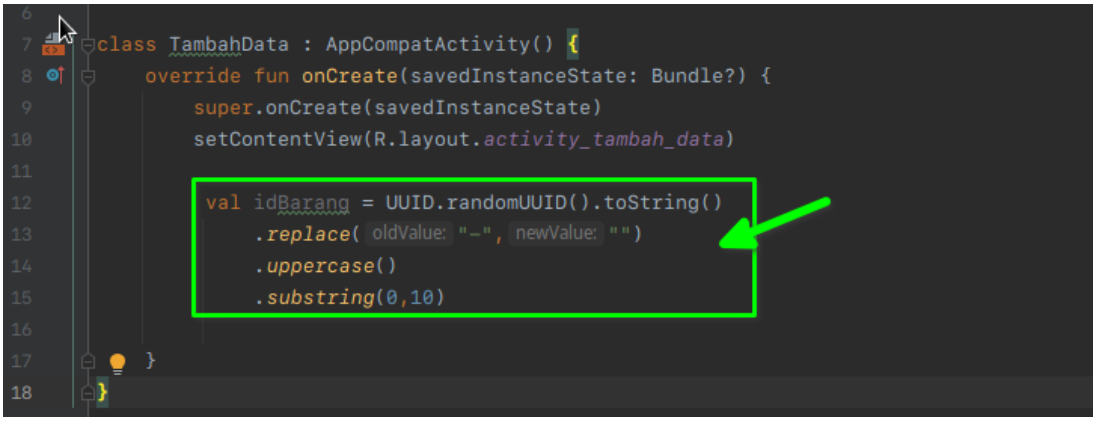


Gambar 1.39: Android : Impor Library Java

29. Setelah itu masukan kode berikut untuk membuat kode **ID Barang** secara unik dan otomatis

Potongan Kode

```
val idBarang = UUID.randomUUID().toString()
        .replace("-","")
        .uppercase()
        .substring(0,10)
```

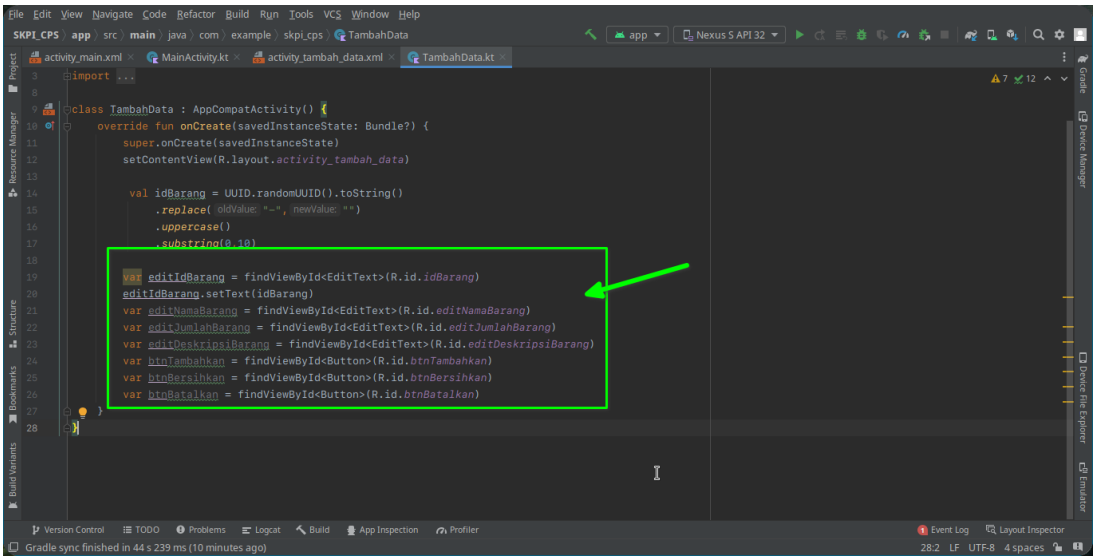


Gambar 1.40: Android : Hasil Kode UUID Generator

30. Masukkan kode berikut untuk menginisialisasi objek-objek yang ada di layout Tambah Data

Potongan Kode

```
var editIdBarang = findViewById<EditText>(R.id.idBarang)
editIdBarang.setText(idBarang)
var editNamaBarang = findViewById<EditText>(R.id.editNamaBarang)
var editJumlahBarang = findViewById<EditText>(R.id.editJumlahBarang)
var editDeskripsiBarang = findViewById<EditText>(R.id.editDeskripsiBarang)
var btnTambahkan = findViewById<Button>(R.id.btnTambahkan)
var btnBersihkan = findViewById<Button>(R.id.btnBersihkan)
var btnBatalkan = findViewById<Button>(R.id.btnBatalkan)
```

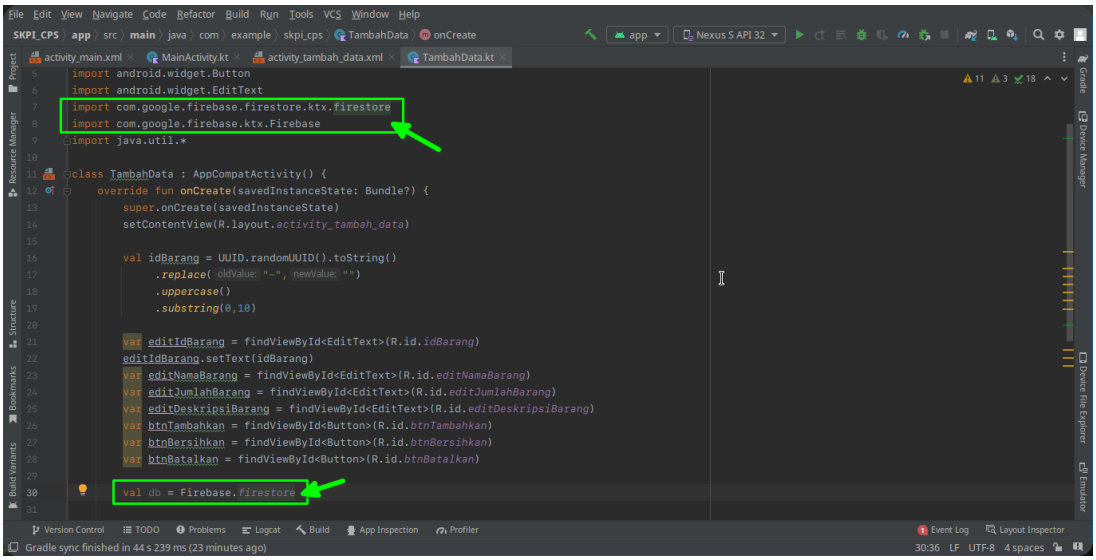


Gambar 1.41: Android : Hasil Kode Inisialisasi

31. Berikutnya adalah menginisalisasi database Firestore dan Import Library nya dengan kode berikut

Potongan Kode

```
val db = Firebase.firestore
```

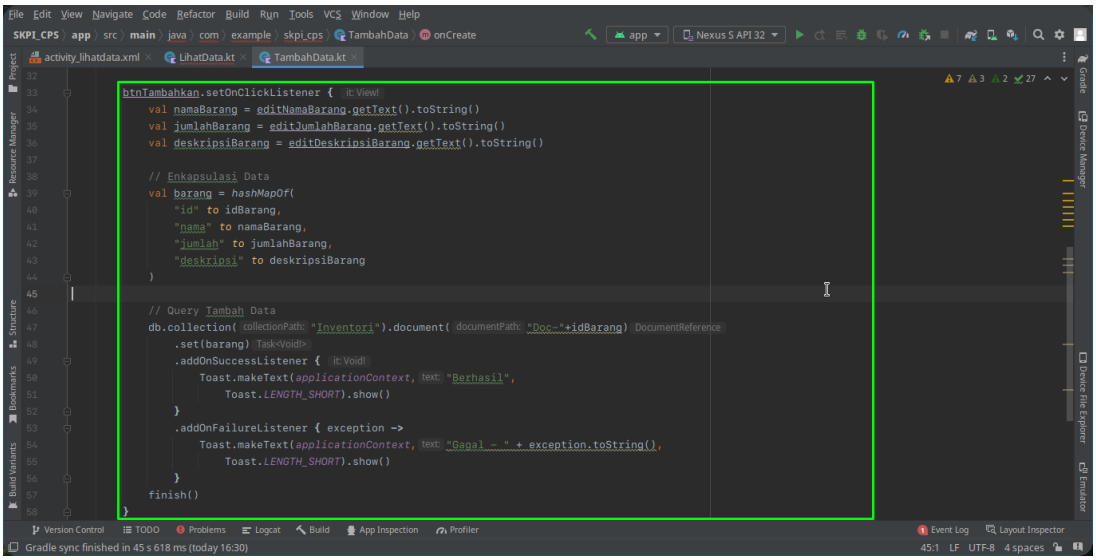


Gambar 1.42: Android : Hasil Kode Akses Database

32. Lalu kode untuk mengambil data dari kolom-kolom yang ada di Layout dan mengirimkannya ke DB. Masukkan kode berikut di bawah kode sebelumnya

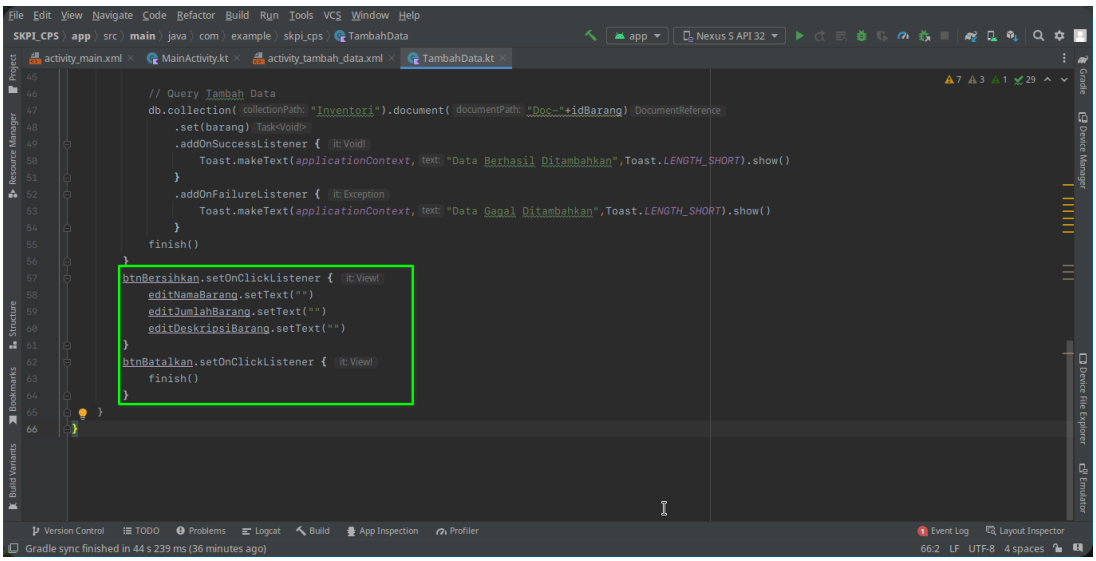
Potongan Kode

```
btnTambahkan.setOnClickListener {
    val namaBarang = editNamaBarang.getText().toString()
    val jumlahBarang = editJumlahBarang.getText().toString()
    val deskripsiBarang = editDeskripsiBarang.getText().toString()

    // Enkapsulasi Data
    val barang = hashMapOf(
        "id" to idBarang,
        "nama" to namaBarang,
        "jumlah" to jumlahBarang,
        "deskripsi" to deskripsiBarang
    )

    // Query Tambah Data
    db.collection("Inventori").document("Doc-"+idBarang)
        .set(barang)
        .addOnSuccessListener {
            Toast.makeText(applicationContext,"Berhasil",
                Toast.LENGTH_SHORT).show()
        }
        .addOnFailureListener { exception ->
            Toast.makeText(applicationContext,"Gagal - " + exception.toString(),
                Toast.LENGTH_SHORT).show()
    }
    finish()
}
btnBersihkan.setOnClickListener {
    editNamaBarang.setText("")
    editJumlahBarang.setText("")
    editDeskripsiBarang.setText("")
}
btnBatalkan.setOnClickListener {
    finish()
}
```



```
btnTambahkan.setOnClickListener {
    val namaBarang = editNamaBarang.getText().toString()
    val jumlahBarang = editJumlahBarang.getText().toString()
    val deskripsiBarang = editDeskripsiBarang.getText().toString()

    // Enkapsulasi Data
    val barang = hashMapOf(
        "id" to idBarang,
        "nama" to namaBarang,
        "jumlah" to jumlahBarang,
        "deskripsi" to deskripsiBarang
    )

    // Query Tambah Data
    db.collection( collectionPath: "Inventori").document( documentPath: "Doc-"+idBarang) DocumentReference
        .set(barang) Task<Void!>
        .addOnSuccessListener { it: Void!
            Toast.makeText(applicationContext, text: "Berhasil",
                Toast.LENGTH_SHORT).show()
        }
        .addOnFailureListener { exception ->
            Toast.makeText(applicationContext, text: "Gagal - " + exception.toString(),
                Toast.LENGTH_SHORT).show()
        }
    finish()
}
```

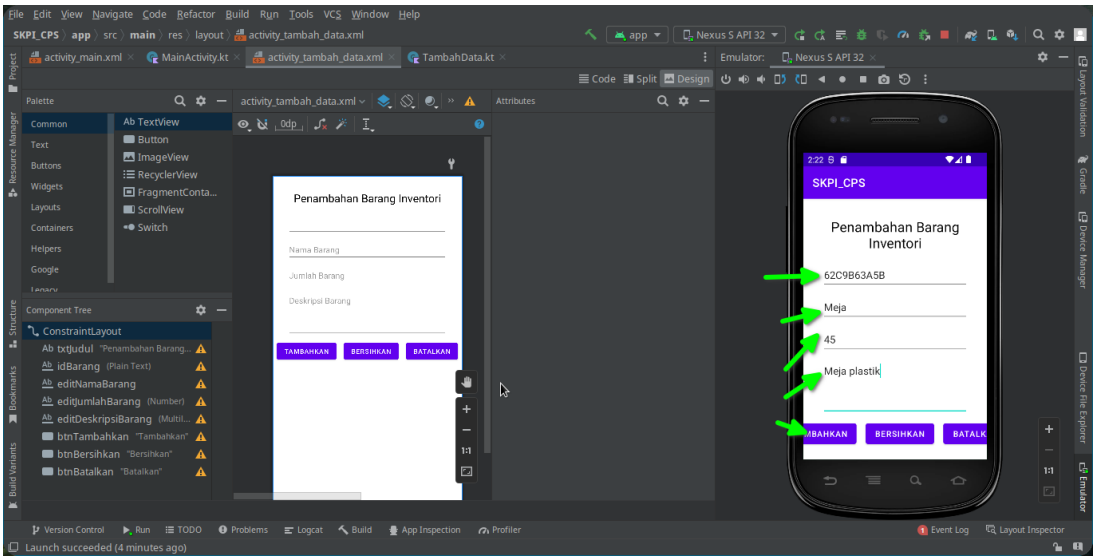
Gambar 1.43: Android : Hasil Kode Button Tambah Data



```
    // Query Tambah Data
    db.collection( collectionPath: "Inventori").document( documentPath: "Doc-"+idBarang) DocumentReference
        .set(barang) Task<Void!>
        .addOnSuccessListener { it: Void!
            Toast.makeText(applicationContext, text: "Data Berhasil Ditambahkan",Toast.LENGTH_SHORT).show()
        }
        .addOnFailureListener { it: Exception
            Toast.makeText(applicationContext, text: "Data Gagal Ditambahkan",Toast.LENGTH_SHORT).show()
        }
    finish()
}
btnBersihkan.setOnClickListener { it: View!
    editNamaBarang.setText("")
    editJumlahBarang.setText("")
    editDeskripsiBarang.setText("")
}
btnBatalkan.setOnClickListener { it: View!
    finish()
}
}
}
```

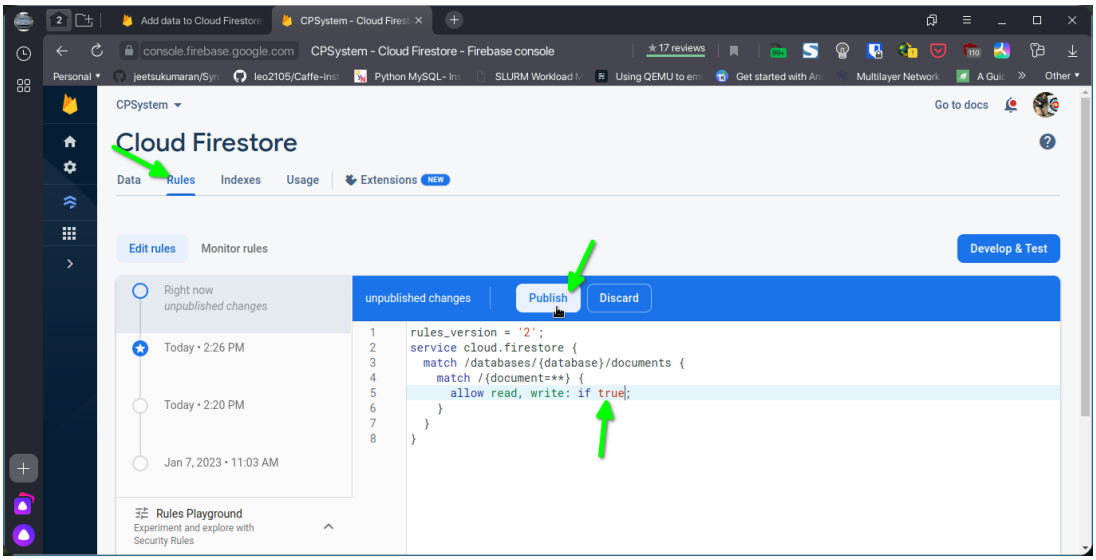
Gambar 1.44: Android : Hasil Kode Bersihkan

33. Jalankan aplikasi dan coba untuk mengirimkan data barang



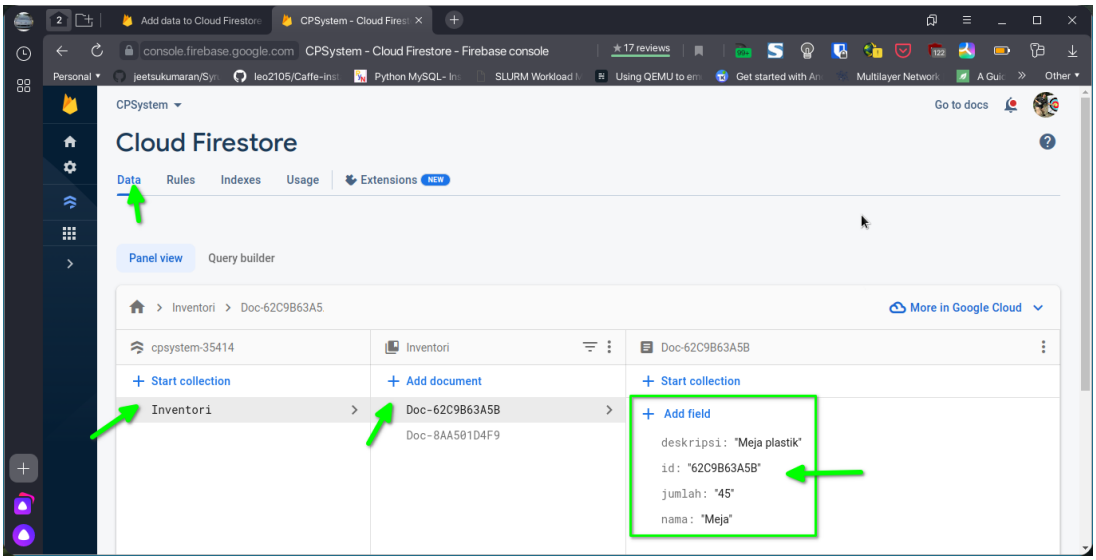
Gambar 1.45: Android : Aplikasi Dijalankan

34. Namun kueri akan gagal karena **Database** masih terkunci. Buka konsole **Firestore** di https://console.firebase.google.com. Buka Database dan klik **Rules**. Ubah **false** menjadi **true** lalu klik **Publish**



Gambar 1.46: Halaman Konfigurasi Keamanan Firebase

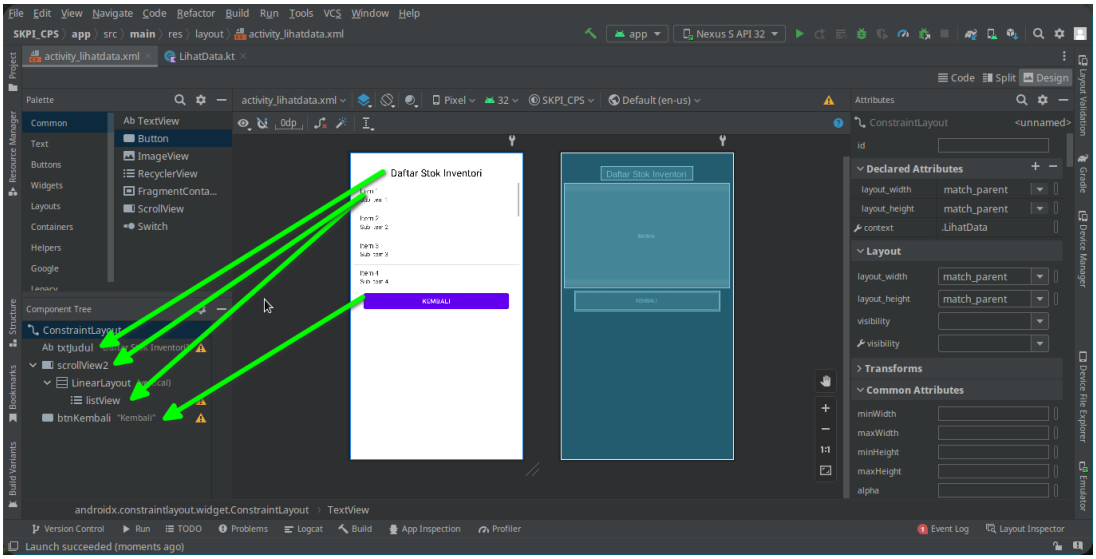
35. Setelah selesai mengkonfigurasi **Rules**, kembali ke aplikasi dan kirim ulang datanya. Data akan muncul di database



Gambar 1.47: Halaman Hasil Kueri Database

1.2.3 Aplikasi Android #2 - Pengambilan Inventori

1. Untuk menampilkan data dari Database, aplikasi Android bisa menggunakan **ListView** untuk menampilkan list data yang ada
2. Buka layout **activity_lihatdata.xml** untuk mengatur tampilan data. Kemudian masukkan objek view seperti berikut:
   - txtJudul - TextView
   - ScrollView (Attributes: **width 250dp**)
   - listView - ListView (Legacy) di dalam ScrollView (Attributes: **width 250dp**)
   - btnKembali - Button

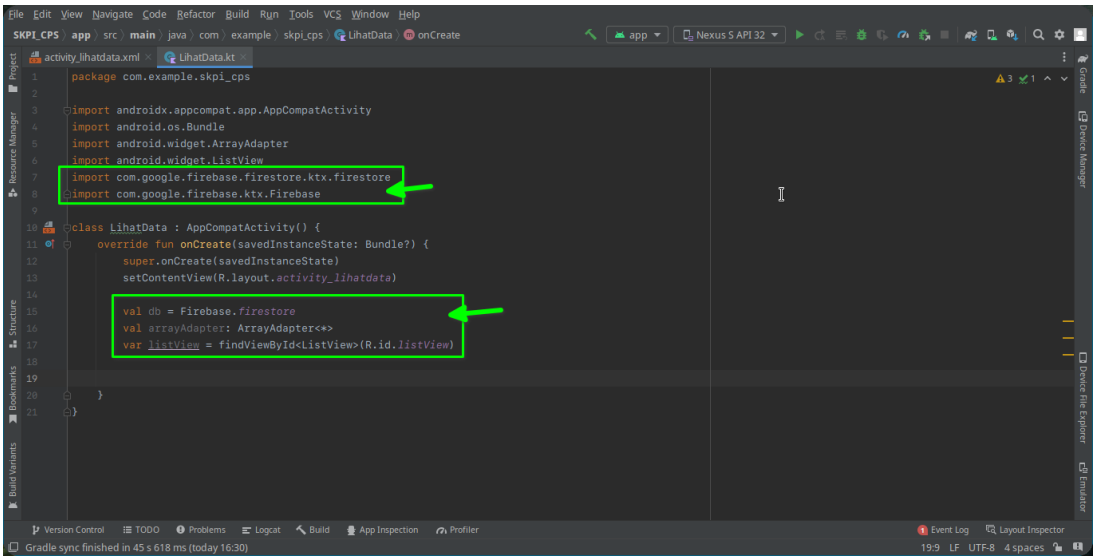


Gambar 1.48: Android : Tampilan Lihat Data

3. Berikutnya adalah memrogram **LihatData**, masukkan kode berikut

Potongan Kode

```
val db = Firebase.firestore
var arrayAdapter: ArrayAdapter<*>
var listView = findViewById<ListView>(R.id.listView)
```

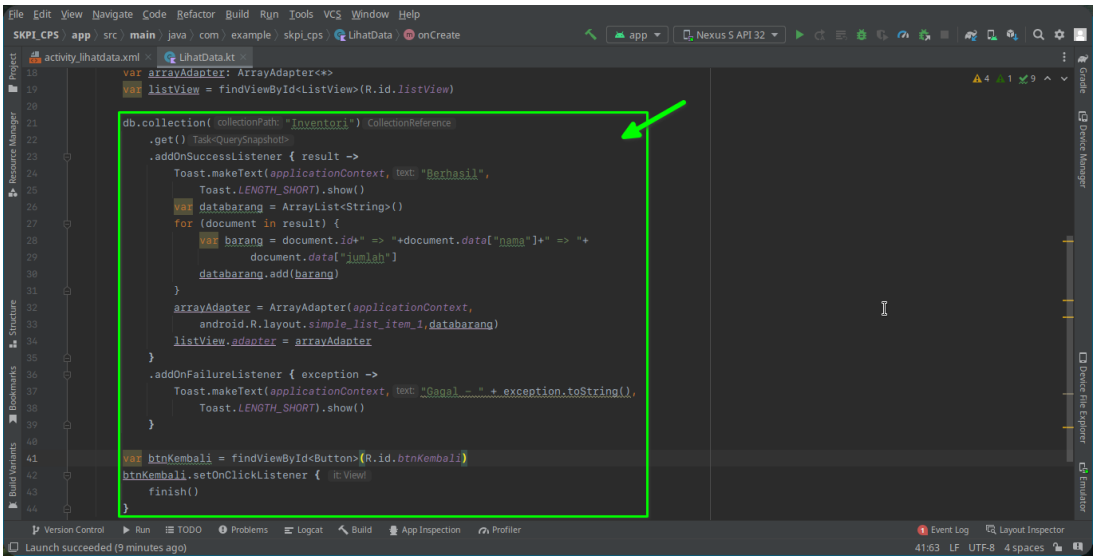


Gambar 1.49: Android : Kode Inisialisasi

4. Jika sudah, masukkan kode pengambil dan penampil data. Lihat kode berikut

Potongan Kode

```
db.collection("Inventori")
    .get()
    .addOnSuccessListener { result ->
        Toast.makeText(applicationContext,"Berhasil",
            Toast.LENGTH_SHORT).show()
        var databarang = ArrayList<String>()
        for (document in result) {
            var barang = document.id+" => "+document.data["nama"]+
                " => "+ document.data["jumlah"]
            databarang.add(barang)
        }
        arrayAdapter = ArrayAdapter(applicationContext,
        android.R.layout.simple_list_item_1,databarang)
        listView.adapter = arrayAdapter
    }
    .addOnFailureListener { exception ->
            Toast.makeText(applicationContext,"Gagal - " +
                exception.toString(), Toast.LENGTH_SHORT).show()
    }

var btnKembali = findViewById<Button>(R.id.btnKembali)
btnKembali.setOnClickListener {
    finish()
}
```

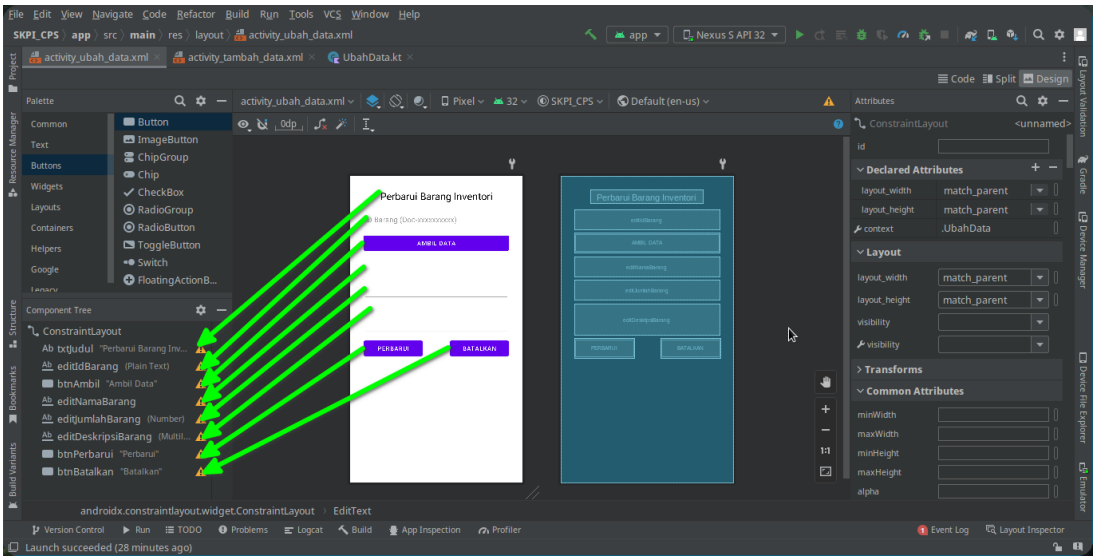


Gambar 1.50: Android : Kode Melihat Data

5. Jalankan Aplikasi dan lihat data yang ditampilkan

1.2.4 Aplikasi Android #3 - Pembaruan Inventori

1. Untuk bisa memperbarui data di database, buka **activity_ubah_data.xml** lalu masukkan komponen berikut, dan perhatikan hasil terakhirnya di gambar
    - txtJudul - TextView
    - editIdBarang - Plain Text
    - btnAmbil - Button
    - editNamaBarang - Plain Text
    - editJumlahBarang - Number
    - editDeskripsiBarang - Multiline Text
    - btnPerbarui - Button
    - btnBatalkan - Button

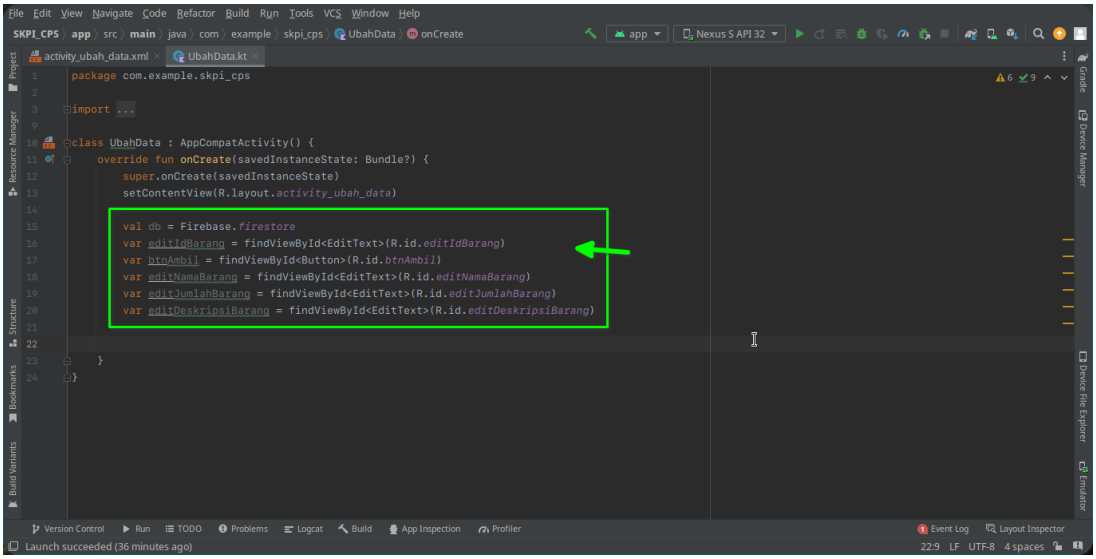


Gambar 1.51: Android : Tampilan Pembaruan Data

2. Agar aplikasi bisa bekerja, masukkan kode berikut dan gambar hasilnya

Potongan Kode

```
val db = Firebase.firestore
var editIdBarang = findViewById<EditText>(R.id.editIdBarang)
var btnAmbil = findViewById<Button>(R.id.btnAmbil)
var editNamaBarang = findViewById<EditText>(R.id.editNamaBarang)
var editJumlahBarang = findViewById<EditText>(R.id.editJumlahBarang)
var editDeskripsiBarang = findViewById<EditText>(R.id.editDeskripsiBarang)
```

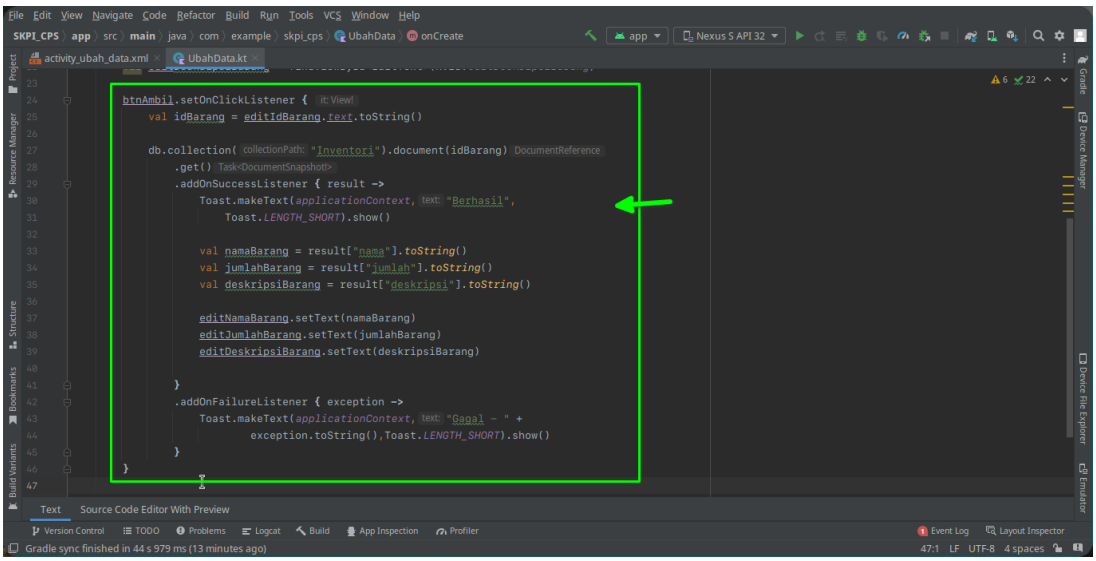


Gambar 1.52: Android : Kode Inisialisasi

3. Berikutnya adalah menambahkan fungsi pengambilan data sesuai dengan Kode Unik Barang. Tambahkan kode berikut

Potongan Kode

```
btnAmbil.setOnClickListener {
    val idBarang = editIdBarang.text.toString()

    db.collection("Inventori").document(idBarang)
        .get()
        .addOnSuccessListener { result ->
            Toast.makeText(applicationContext,"Berhasil",
                Toast.LENGTH_SHORT).show()

            editIdBarang.isEnabled = false
            val namaBarang = result["nama"].toString()
            val jumlahBarang = result["jumlah"].toString()
            val deskripsiBarang = result["deskripsi"].toString()

            editNamaBarang.setText(namaBarang)
            editJumlahBarang.setText(jumlahBarang)
            editDeskripsiBarang.setText(deskripsiBarang)
        }
        .addOnFailureListener { exception ->
            Toast.makeText(applicationContext,"Gagal - " +
                exception.toString(),Toast.LENGTH_SHORT).show()
        }
}
```

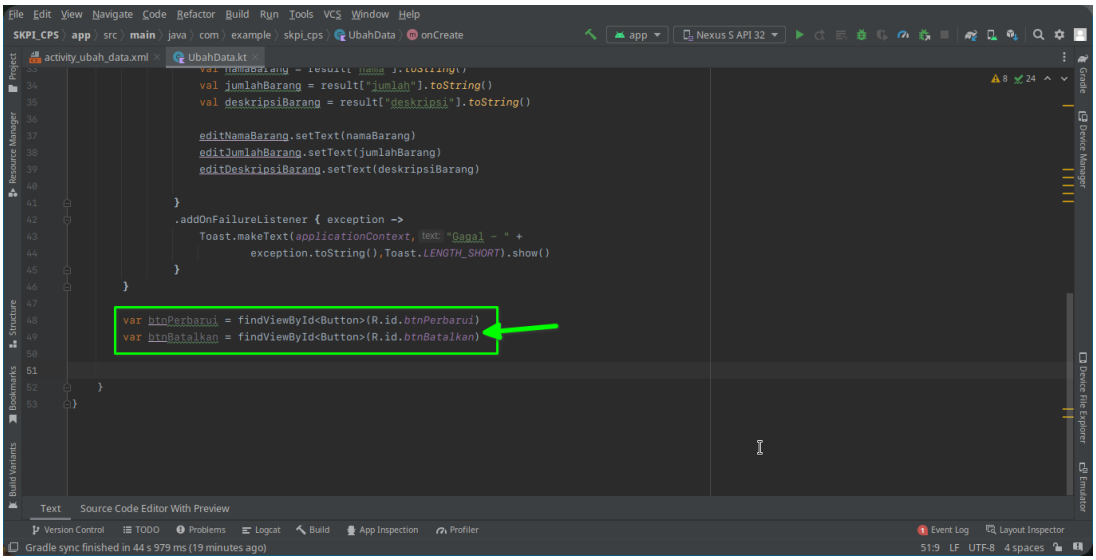


Gambar 1.53: Android : Kode Pengambil Data

4. Berikutnya adalah memasukkan kode untuk pembaruan data. Perhatikan gambar dan kode berikut

Potongan Kode

```
var btnPerbarui = findViewById<Button>(R.id.btnPerbarui)
var btnBatalkan = findViewById<Button>(R.id.btnBatalkan)
```

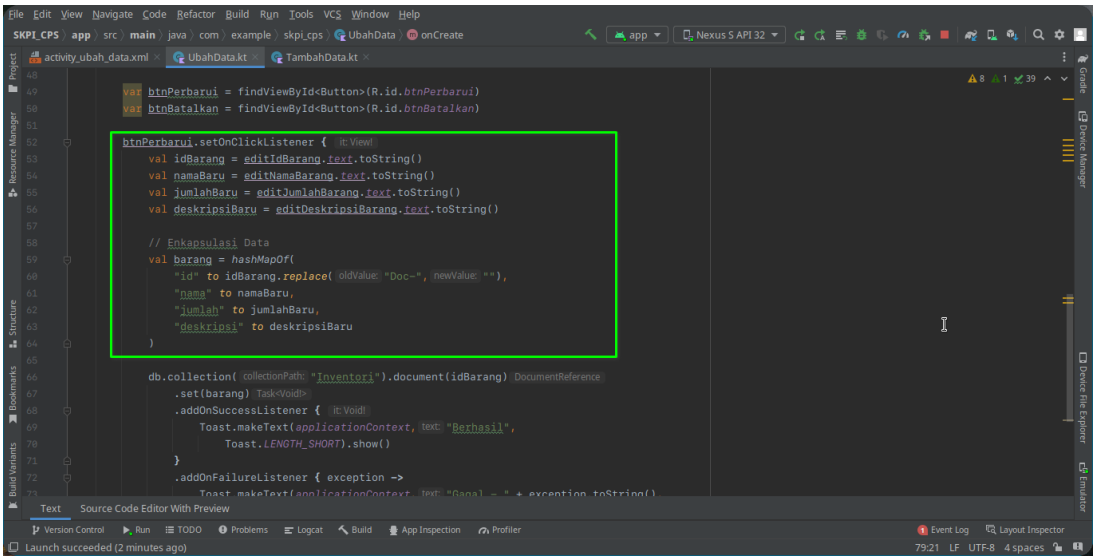


Gambar 1.54: Android : Kode Inisialisasi Tombol

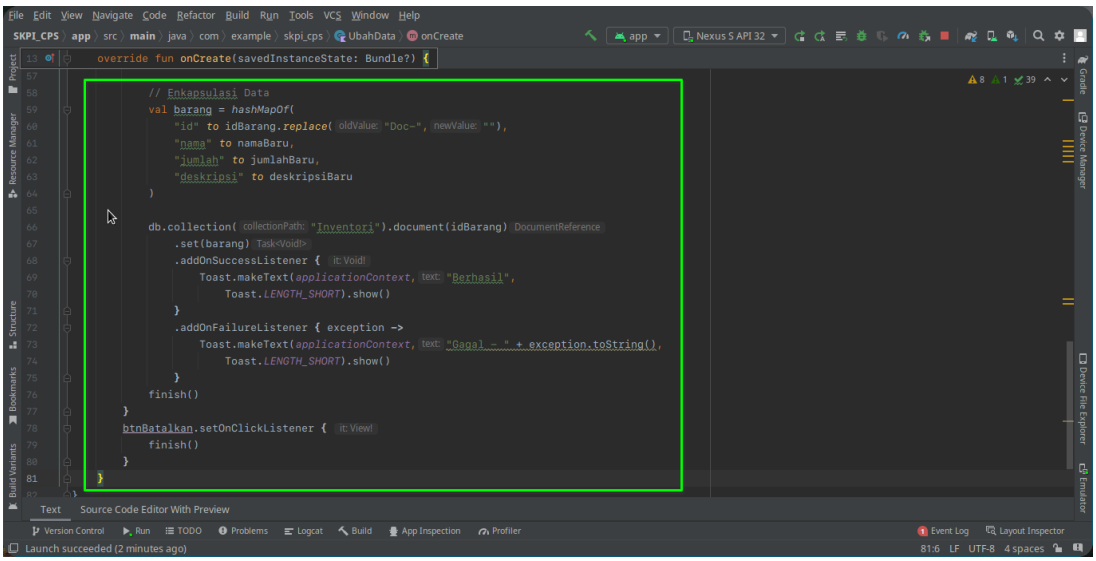
5. Kemudian dilanjutkan dengan fungsi update ke dalam tombol

Potongan Kode

```
btnPerbarui.setOnClickListener {
    val idBarang = editIdBarang.text.toString()
    val namaBaru = editNamaBarang.text.toString()
    val jumlahBaru = editJumlahBarang.text.toString()
    val deskripsiBaru = editDeskripsiBarang.text.toString()

    // Enkapsulasi Data
    val barang = hashMapOf(
        "id" to idBarang.replace("Doc-",""),
        "nama" to namaBaru,
        "jumlah" to jumlahBaru,
        "deskripsi" to deskripsiBaru
    )

    db.collection("Inventori").document(idBarang)
        .set(barang)
        .addOnSuccessListener {
            Toast.makeText(applicationContext,"Berhasil",
                Toast.LENGTH_SHORT).show()
        }
        .addOnFailureListener { exception ->
            Toast.makeText(applicationContext,"Gagal - " +
                exception.toString(),Toast.LENGTH_SHORT).show()
        }
        finish()
}
btnBatalkan.setOnClickListener {
    finish()
}
```

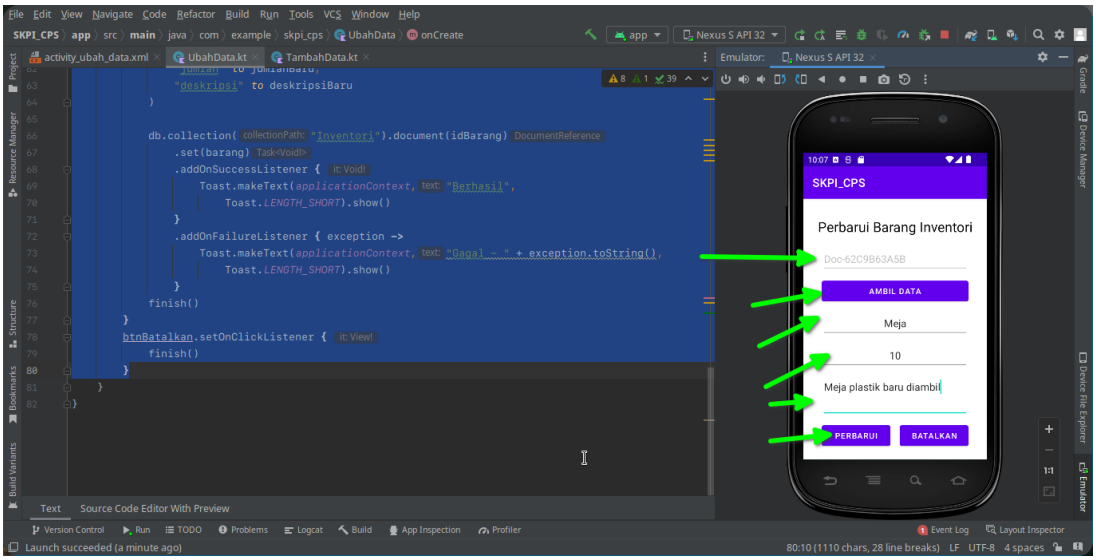


Gambar 1.55: Android : Kode Pembaruan Data #1

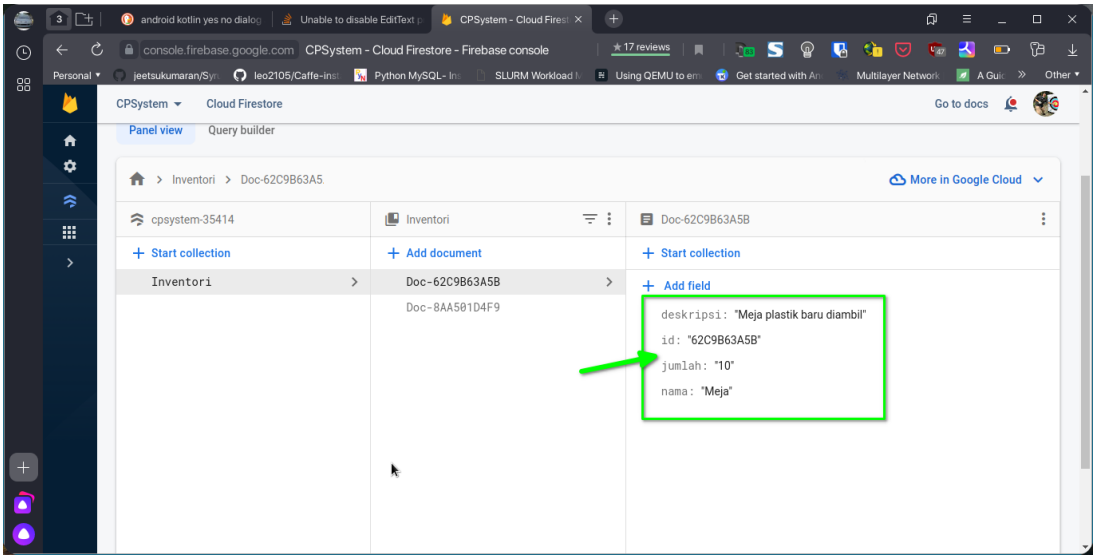


Gambar 1.56: Android : Kode Pembaruan Data #2

6. Jalankan aplikasi, masukkan **Kode Barang**, dan ubahlah data yang ada di database



Gambar 1.57: Android : Aplikasi Membarui Data

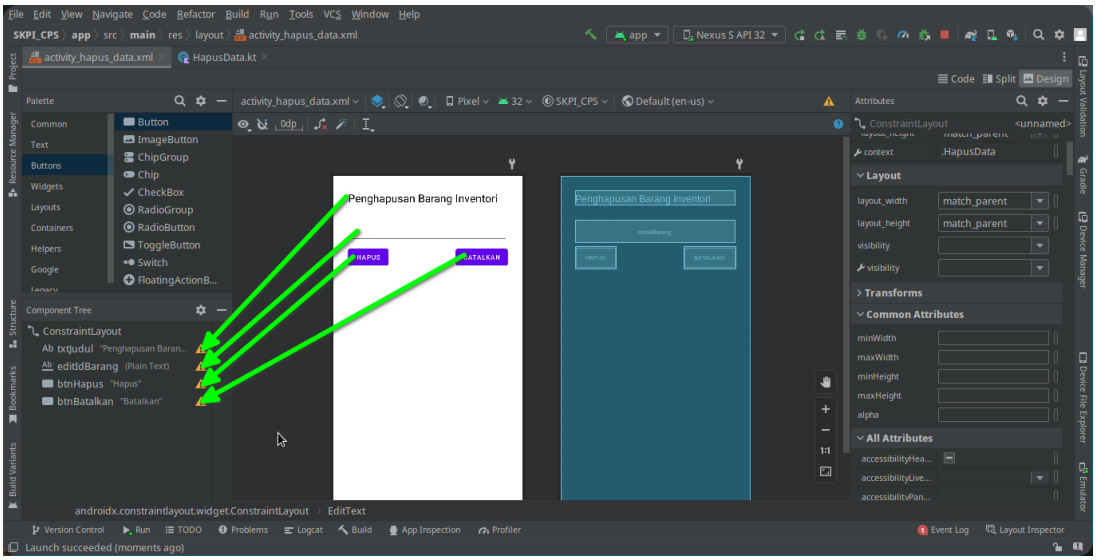


Gambar 1.58: Android : Hasil Pembaruan

1.2.5 Aplikasi Android #4 - Hapus Barang

1. Bagian terakhir adalah menghapus data. Hal ini bisa dilakukan dengan mudah, cukup menggunakan ID Barang untuk menghapus data tersebut.

2. Buka layout **activity_hapus_data.xml** dan atur tampilan seperti berikut:

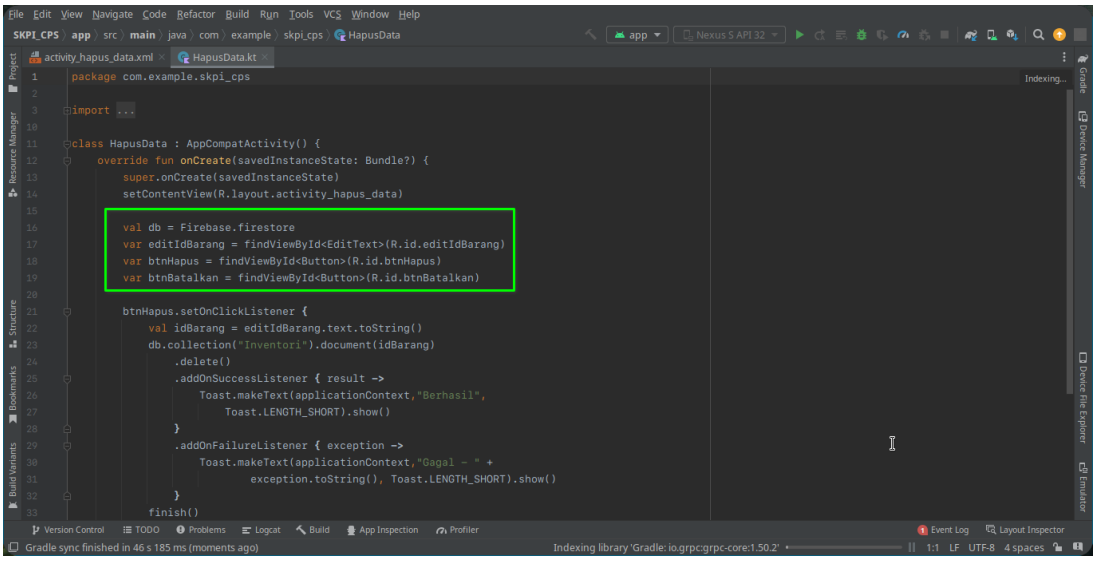


Gambar 1.59: Android : Tampilan Hapus Data

3. Kemudian tambahkan kode Alert Dialog untuk konfirmasi penghapusan terakhir. Perhatikan kode berikut ini

Potongan Kode

```
val db = Firebase.firestore
var editIdBarang = findViewById<EditText>(R.id.editIdBarang)
var btnVerifikasi = findViewById<Button>(R.id.btnVerifikasi)
var btnHapus = findViewById<Button>(R.id.btnHapus)
var btnBatalkan = findViewById<Button>(R.id.btnBatalkan)
```

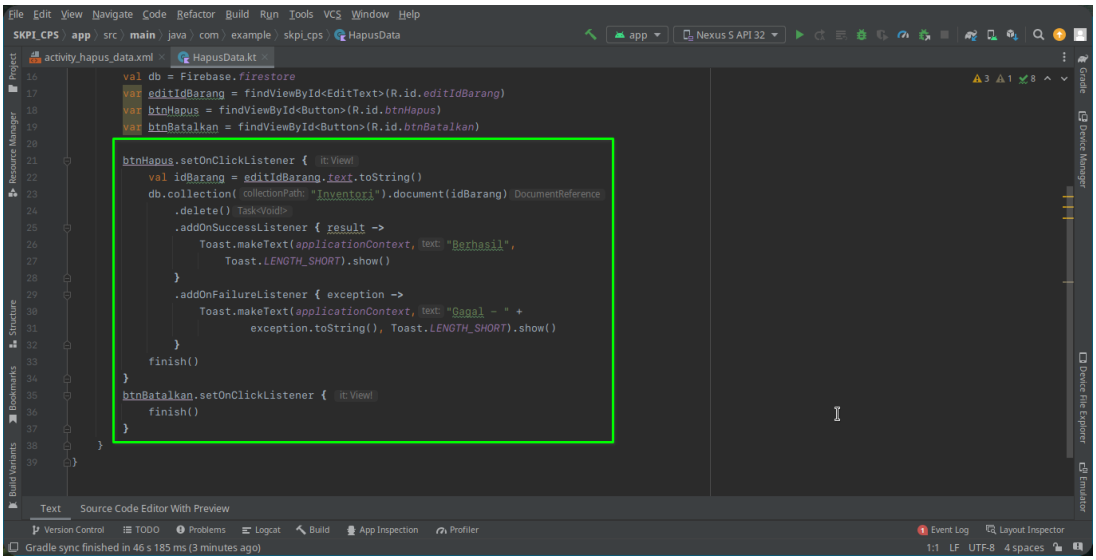


Gambar 1.60: Android : Kode Inisialisasi Tombol

4. Bagian berikutnya adalah sebagai berikut

Potongan Kode

```
btnHapus.setOnClickListener {
    val idBarang = editIdBarang.text.toString()
    db.collection("Inventori").document(idBarang).delete()
        .addOnSuccessListener { result ->
            Toast.makeText(applicationContext,"Berhasil",
                Toast.LENGTH_SHORT).show()
        }
        .addOnFailureListener { exception ->
            Toast.makeText(applicationContext,"Gagal - " +
                exception.toString(), Toast.LENGTH_SHORT).show()
        }
    finish()
}
btnBatalkan.setOnClickListener {
    finish()
}
```



Gambar 1.61: Android : Kode Tombol Hapus Data

5. Jalankan aplikasi dan coba hapus satu dokumen dari database.

Bab 2

Web Application

2.1 Flask Python + Cloud Firebase

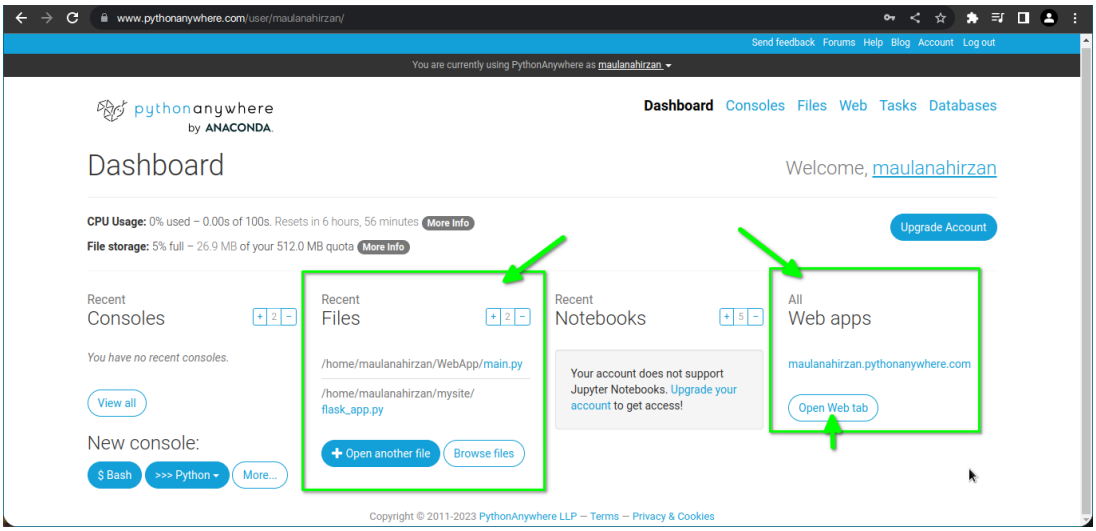
Di bagian ini mahasiswa diajarkan bagaimana membuat projek Web App dengan menggunakan Flask Python dan Cloud Firestore. Mahasiswa diwajibkan memiliki **Akun Google** dan **pythonanywhere.com**

2.2 Tutorial

2.2.1 WebApp #1 - Home

1. Sebelum membuat WebApp, sebaiknya mahasiswa membuat akun di website `https://www.pythonanywhere.com` secara gratis.

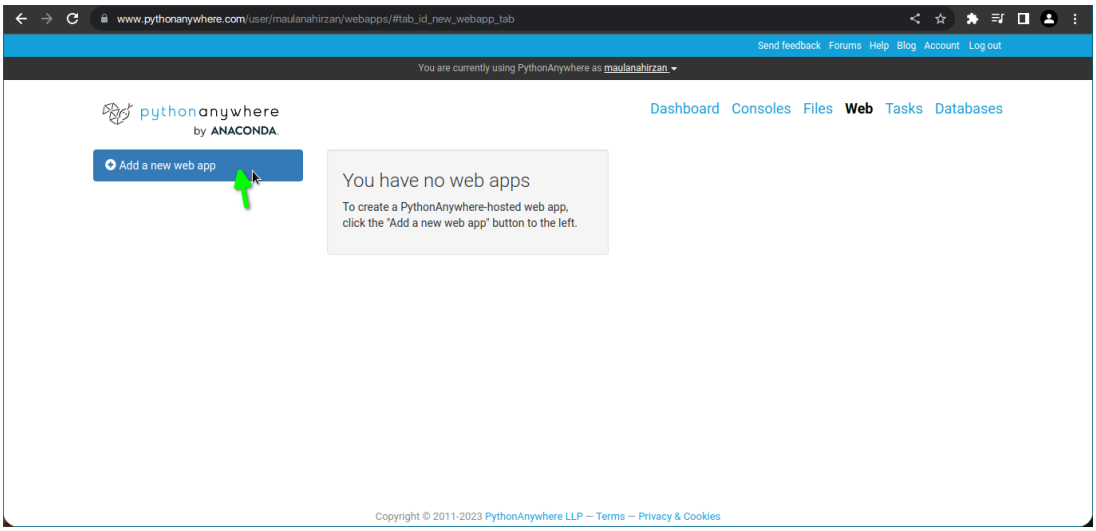
2. Jika sudah selesai mendaftar dan login, tampilan akan menjadi seperti di bawah. Ada dua bagian penting **Web apps** untuk membuat dan menjalankan Web App dan **Files** untuk mengakses file python (File Manager). Klik **Open Web tab**



Gambar 2.1: WebApp : Tampilan PaaS PythonAnywhere

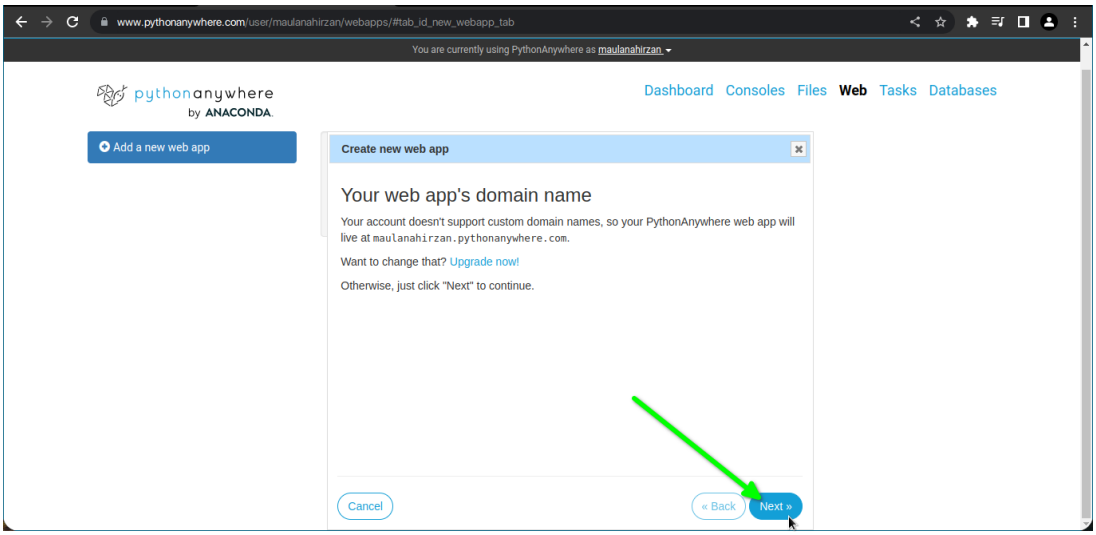
3. PaaS akan membuka halaman untuk membuat WebApps. Klik **Add a new web**

app



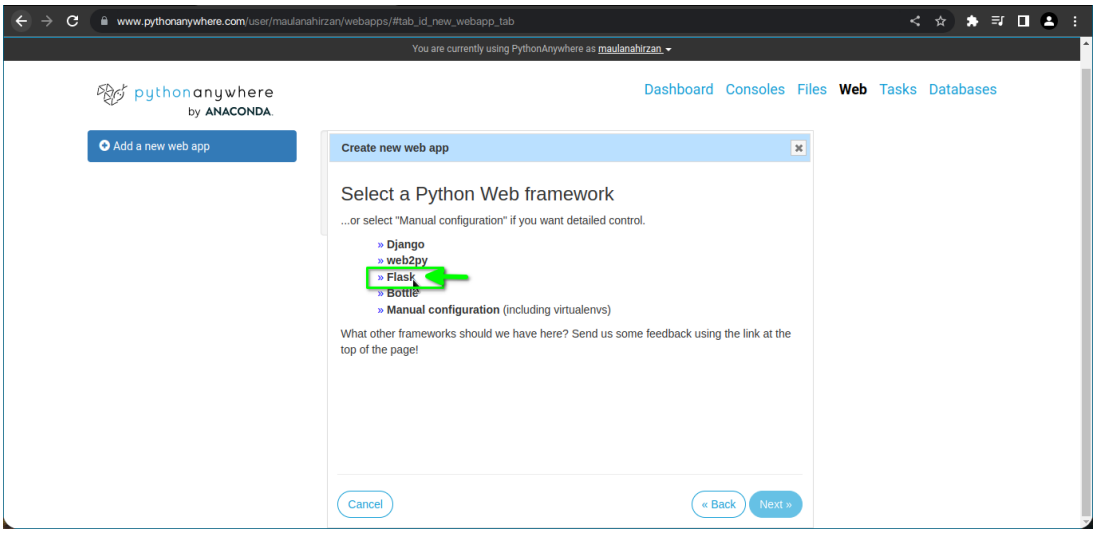
Gambar 2.2: WebApp : Membuat WebApp Baru

4. Klik **Next** untuk nama domain default dari website.

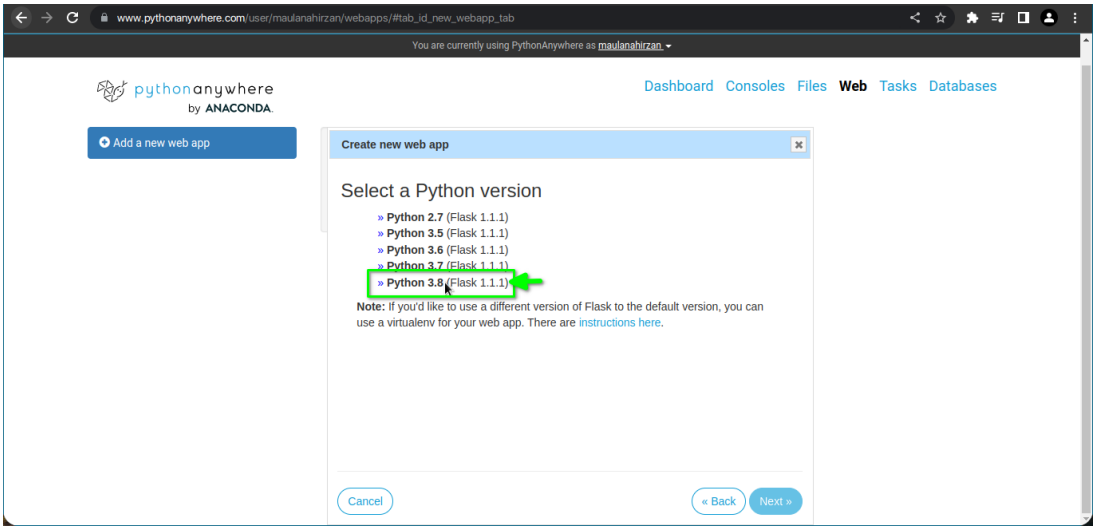


Gambar 2.3: WebApp : Nama Domain Default

5. Pilih **Flask** lalu **Python 3.8**

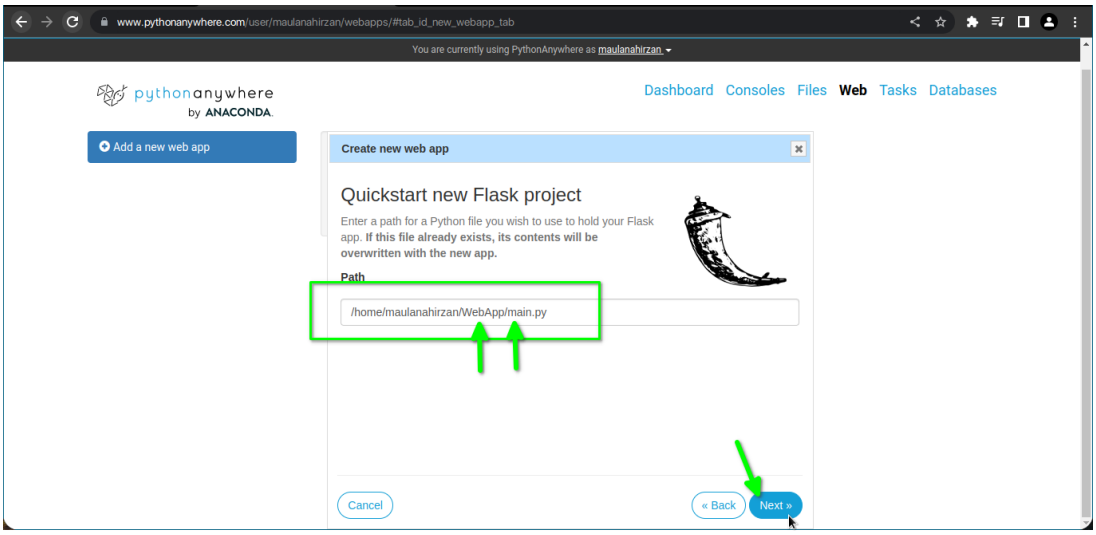


Gambar 2.4: WebApp : Memilih Framework Web Python



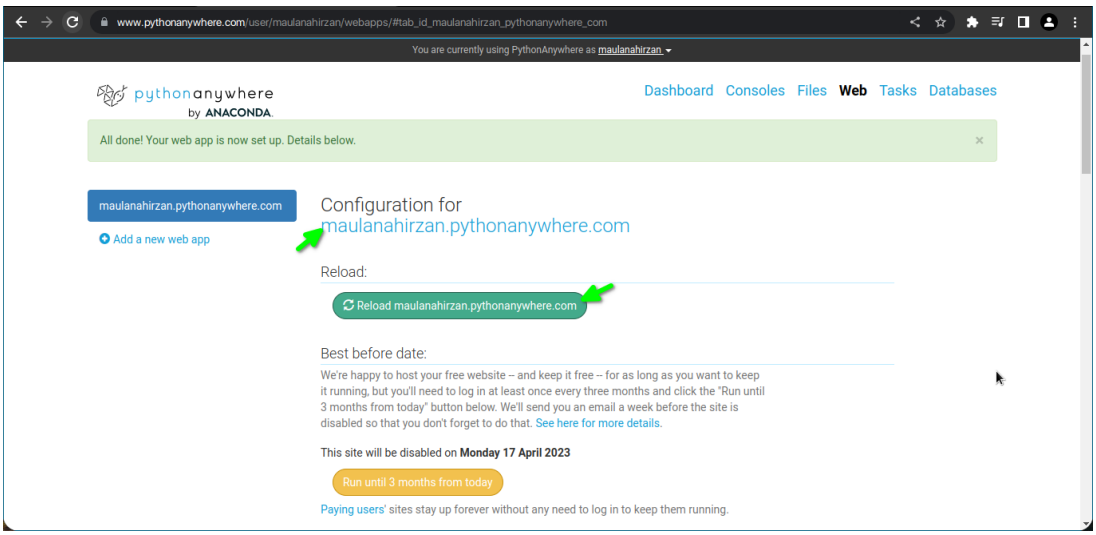
Gambar 2.5: WebApp : Memilih Versi Python

6. Ubah target folder **mysite** menjadi **WebApp**, dan nama file **flask_app.py** menjadi **main.py**. Lalu klik **Next**



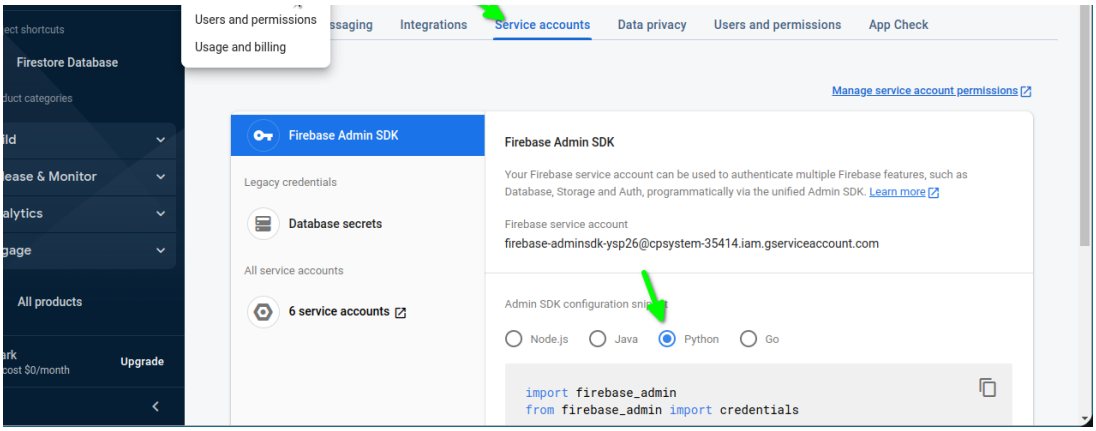
Gambar 2.6: WebApp : Mengubah Folder Default dan File

7. **WebApp** sudah aktif dan bisa diakses melalui browser. Untuk setiap perubahan file cukup klik **Reload**



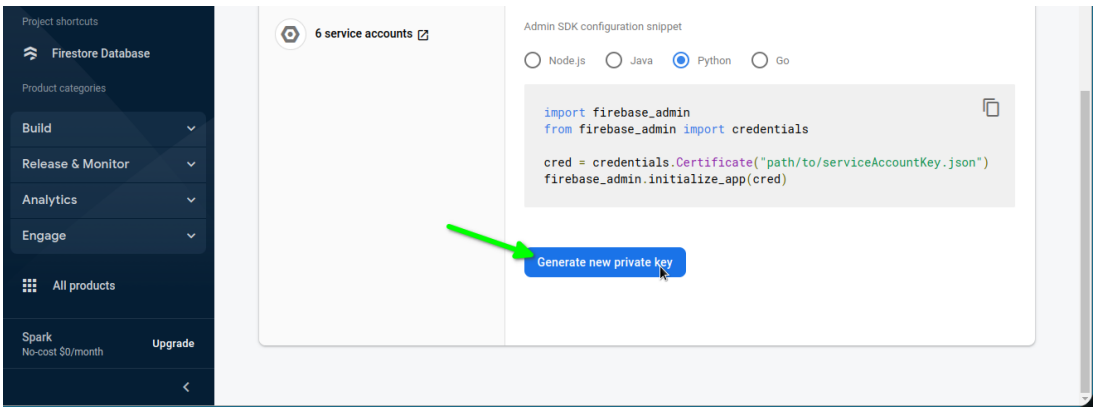
Gambar 2.7: WebApp : Link Web dan Tombol Reload

8. Setelah website siap, berikutnya adalah mengunggah file otorisasi database. Namun sebelumnya buka `https://console.firebase.google.com` => Buka Projek.
9. Untuk mengunduh file otorisasi klik **Roda Gigi** => **Project Settings** => **Service Account** => Pilih **Python**

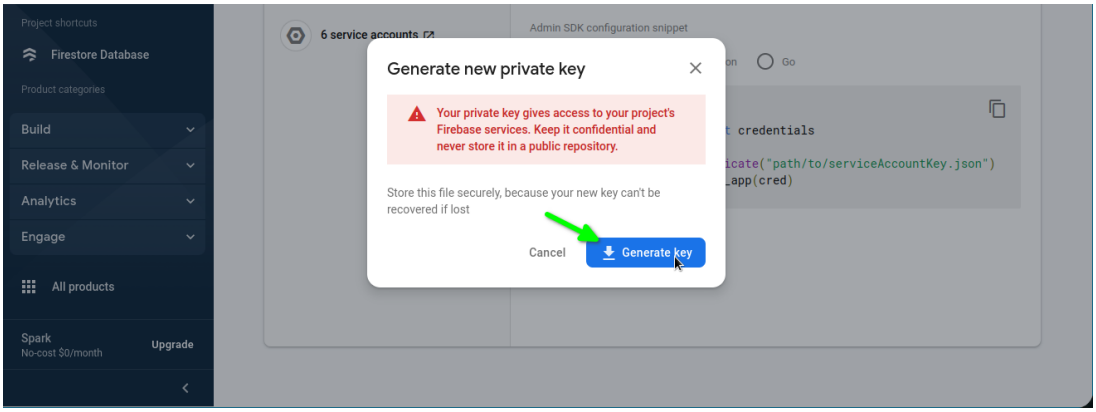


Gambar 2.8: WebApp : Project Settings untuk Python

10. Klik **Generate Key** dua kali. Maka **Browser** akan mengunduh file **JSON**

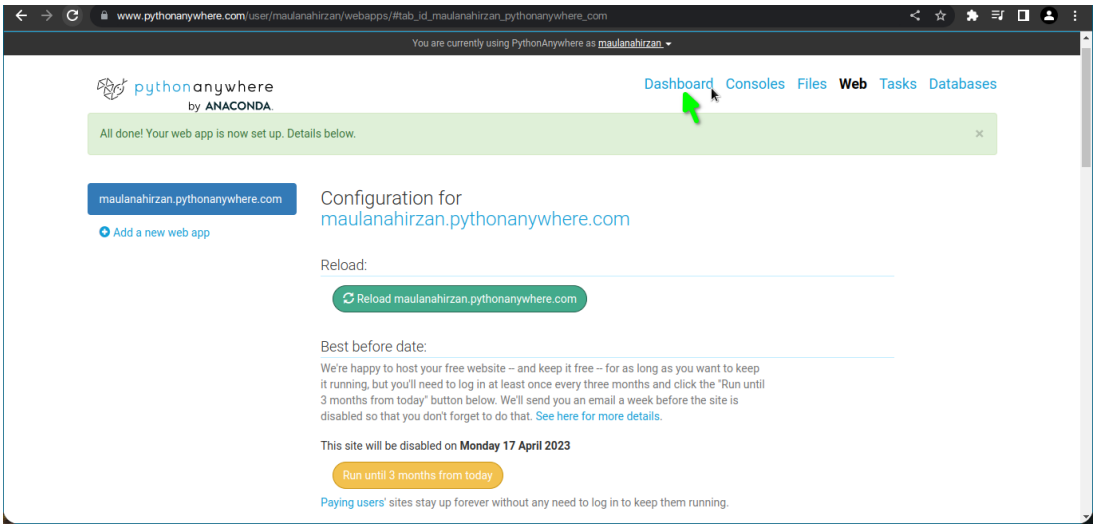


Gambar 2.9: WebApp : Generate Key #1



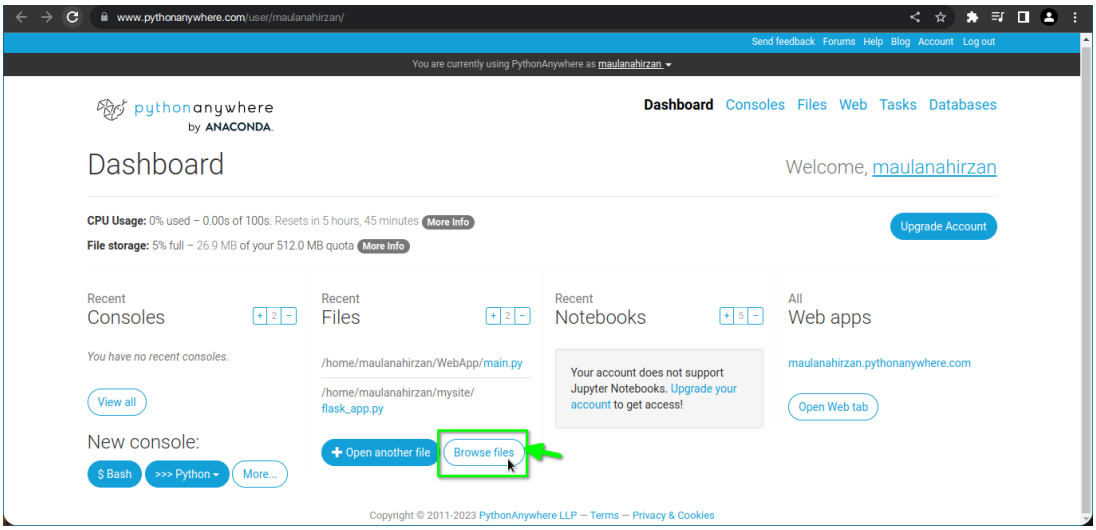
Gambar 2.10: WebApp : Generate Key #2

11. Ganti nama file **JSON** tersebut menjadi **cpsystem-adminsdk.json**.
12. Untuk mengunggah file **JSON**. Buka **Dashboard** di **Tab baru**



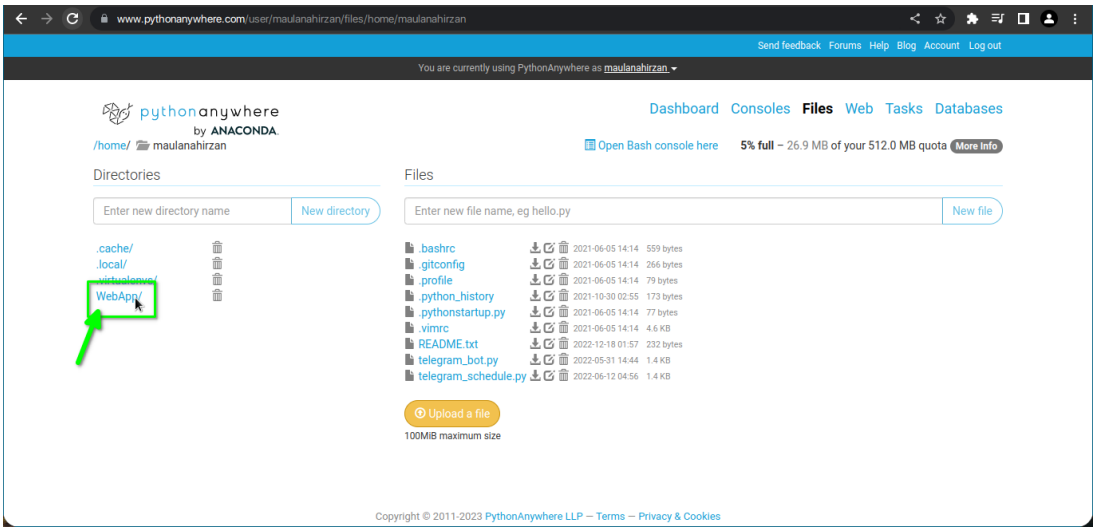
Gambar 2.11: WebApp : Mengakses Dashboard

13. Klik **Browse files**



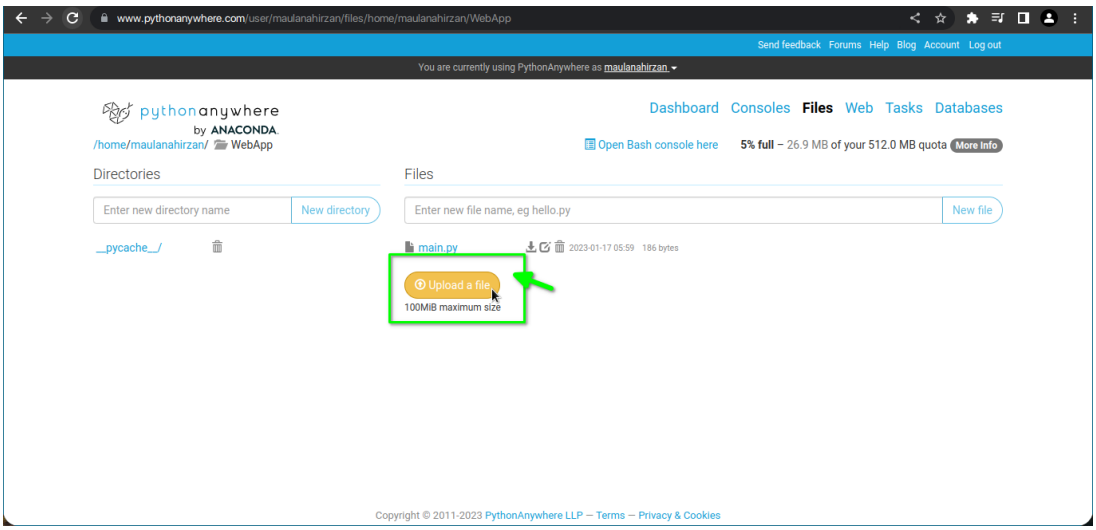
Gambar 2.12: WebApp : Mengakses Files

14. Klik Folder **WebApp**



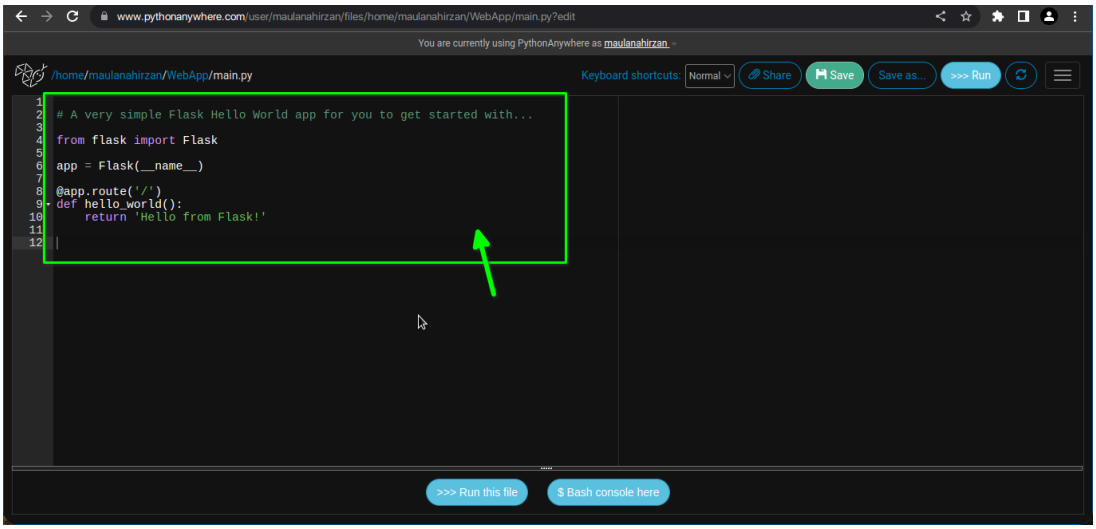
Gambar 2.13: WebApp : Membuka Folder WebApp

15. Klik **Upload a file**



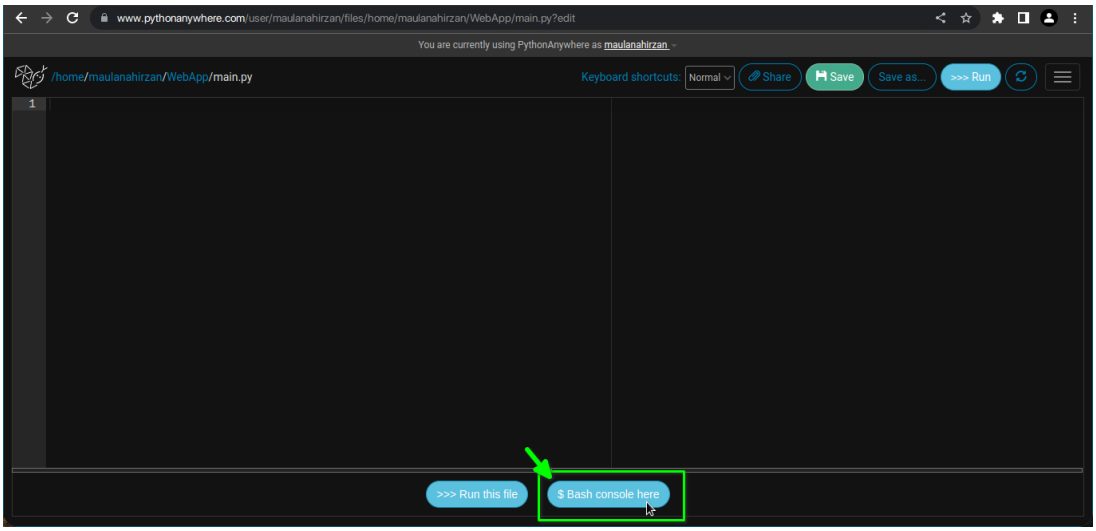
Gambar 2.14: WebApp : Mengunggah File

16. Jika file sudah terunggah, buka **main.py** yang ada di **Files** dan mulai memrogram **WebApp**. Hapus semua baris kode yang ada



Gambar 2.15: WebApp : Hapus Kode Lama

17. Sebelum memulai memrogram, klik **Bash console here**

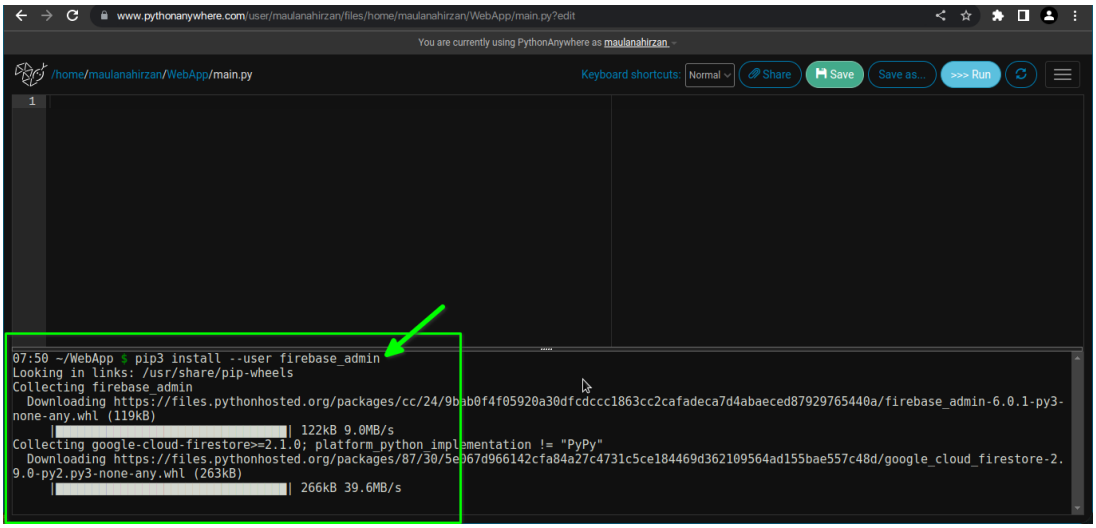


Gambar 2.16: WebApp : Membuka Bash Console

18. PaaS akan membuka **console** di bagian bawah. Lalu masukkan perintah berikut ke **console** dan tunggu hingga selesai

Perintah Terminal

```
pip3 install --user firebase_admin==3.2.1
```

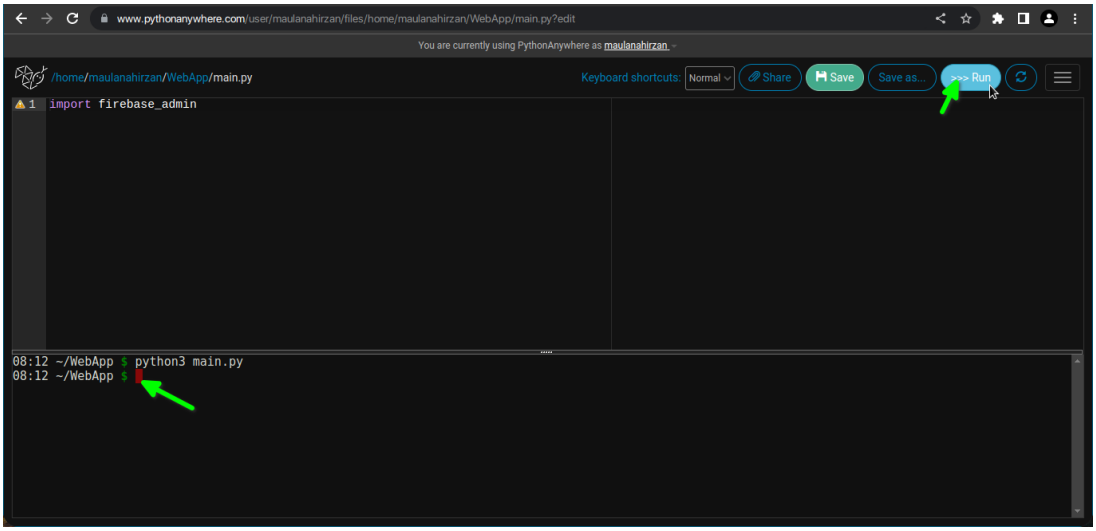


Gambar 2.17: WebApp : Instalasi firebase_admin

19. Untuk menguji apakah library sudah terinstall dengan baik, masukkan potongan kode berikut, dan klik **Run**

Potongan Kode

```
import firebase_admin
```

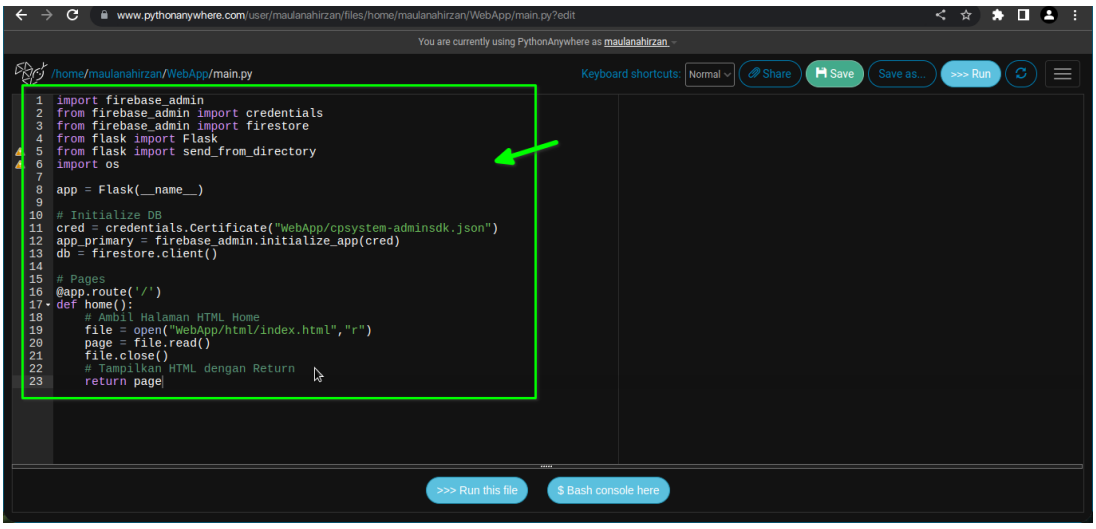


Gambar 2.18: WebApp : Hasil Ketika Dijalankan

20. Masukkan kode berikut untuk dapat menampilkan **Home**. Perhatikan Indentasi, gunakan **4 SPASI** bukan **Tab**

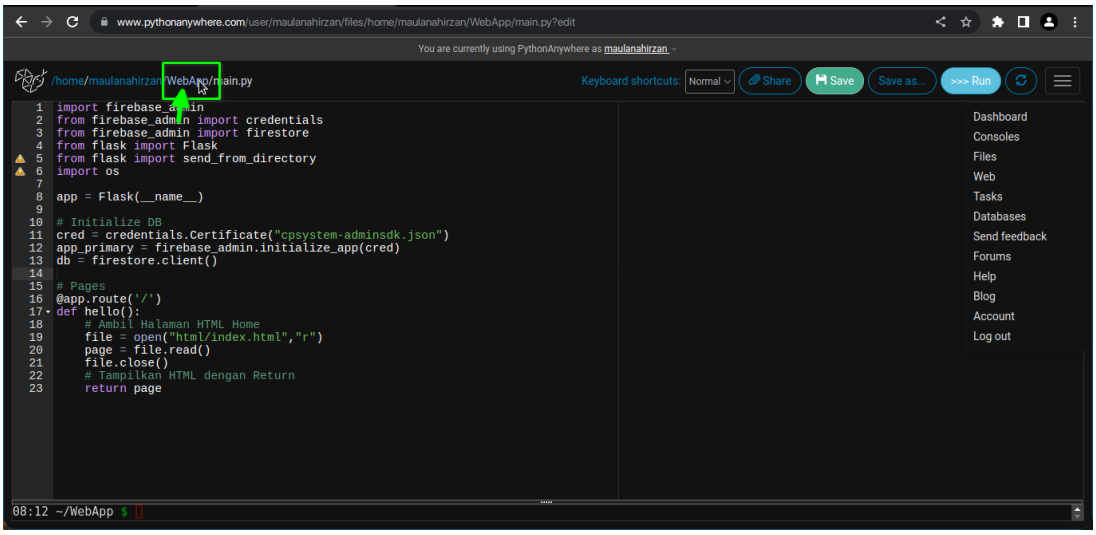
Potongan Kode

```
import firebase_admin
from firebase_admin import credentials
from firebase_admin import firestore
from flask import Flask
from flask import send_from_directory
import os

app = Flask(__name__)

# Initialize DB
cred = credentials.Certificate("WebApp/cpsystem-adminsdk.json")
app_primary = firebase_admin.initialize_app(cred)
db = firestore.client()

# Pages
@app.route('/')
def home():
    # Ambil Halaman HTML Home
    file = open("WebApp/html/index.html","r")
    page = file.read()
    file.close()
    # Tampilkan HTML dengan Return
    return page
```

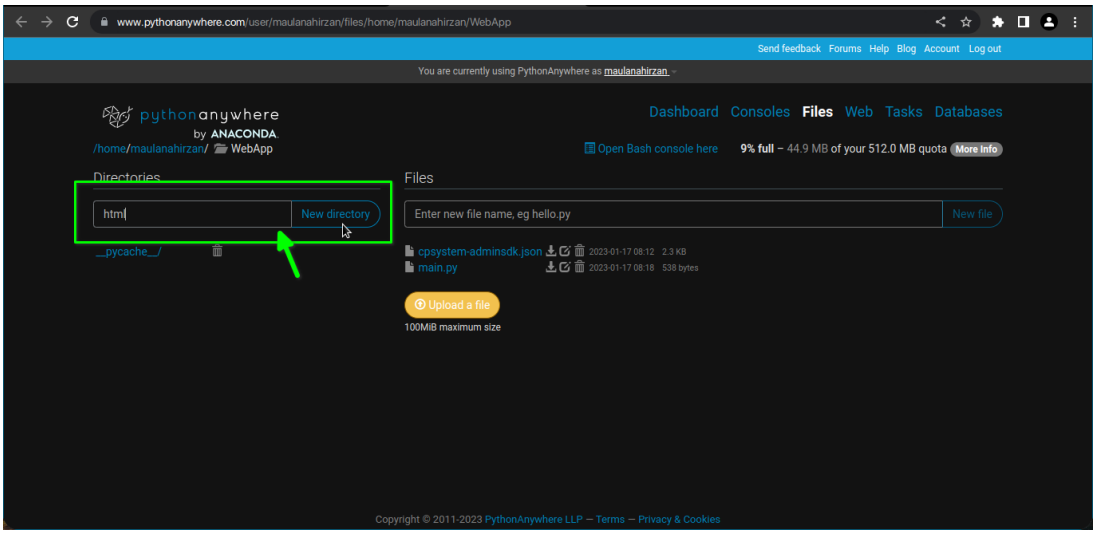


Gambar 2.19: WebApp : Kode Halaman Awal WebApp

21. Kode ini akan memanggil file **index.html** di folder **html**. Buka halaman **Folder WebApp** di **Tab Baru** dan buat folder **html** di dalam folder **WebApp**

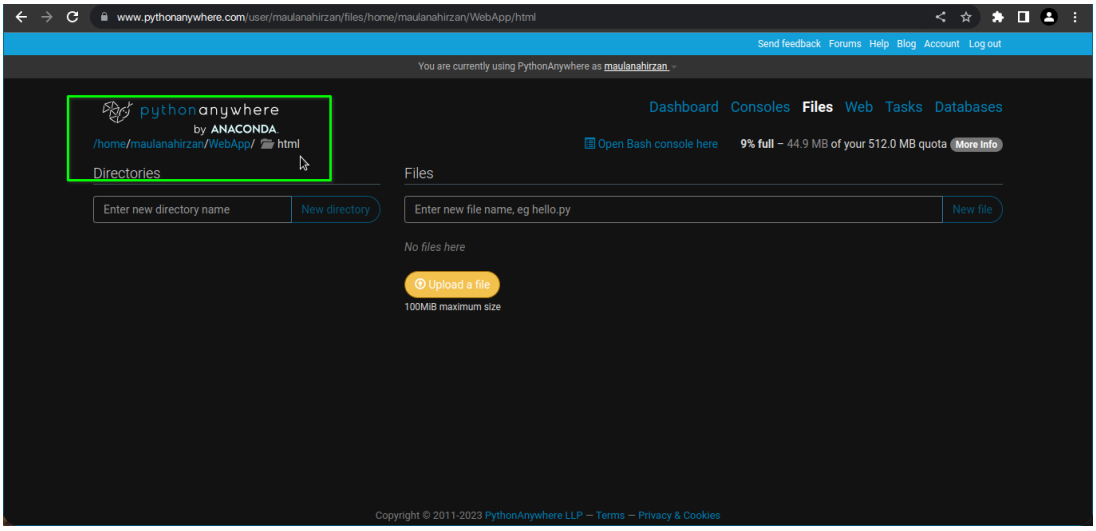


Gambar 2.20: WebApp : Mengakses Folder WebApp



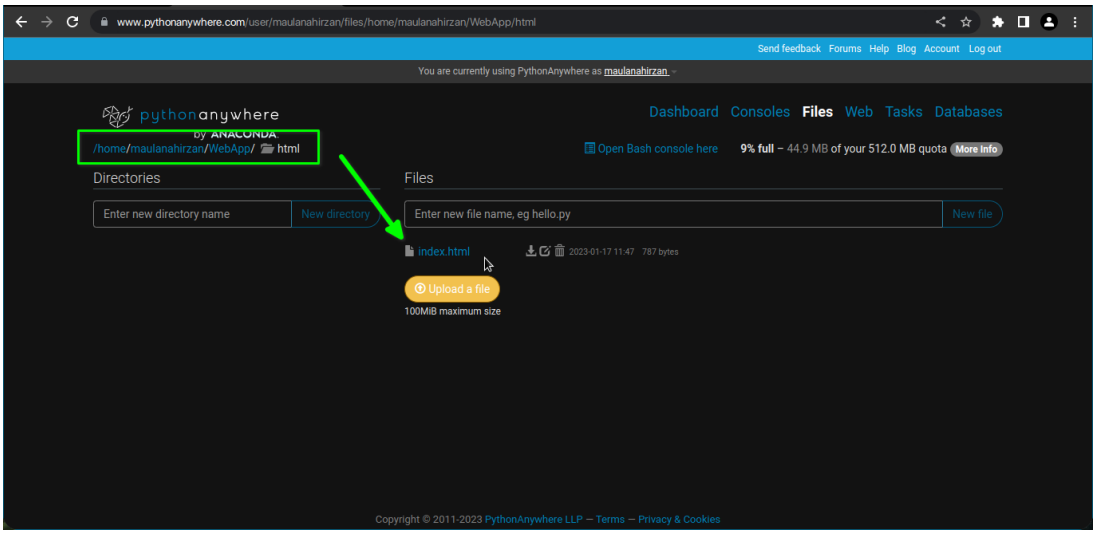
Gambar 2.21: WebApp : Membuat Folder HTML

22. Folder **html** ini digunakan untuk meletakkan file-file **HTML**.



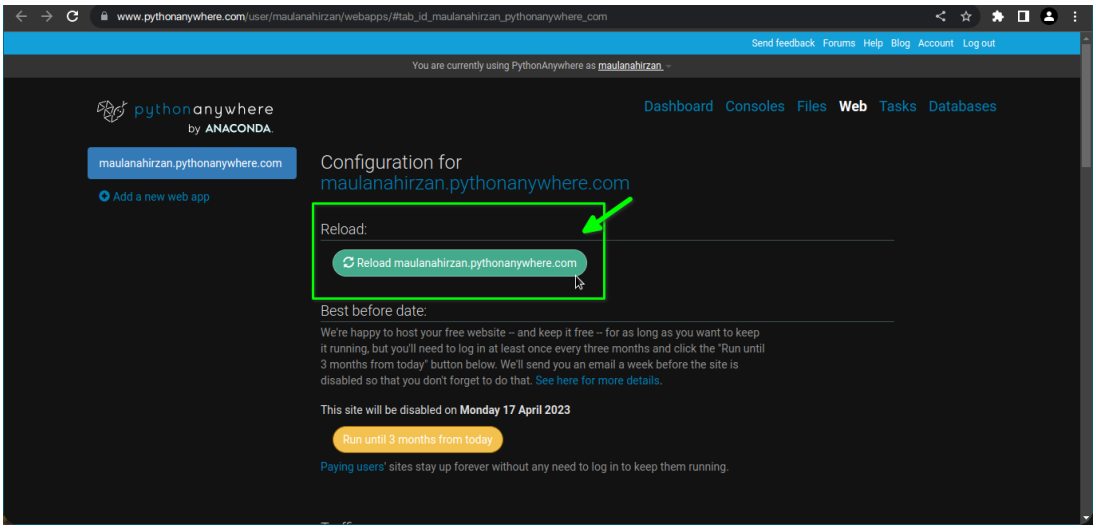
Gambar 2.22: WebApp : Mengecek Lokasi Folder

23. Unduh file HTML melalui link yang sudah disediakan di Bagian 0.7, dan unggah file **index.html** ke dalam folder tersebut



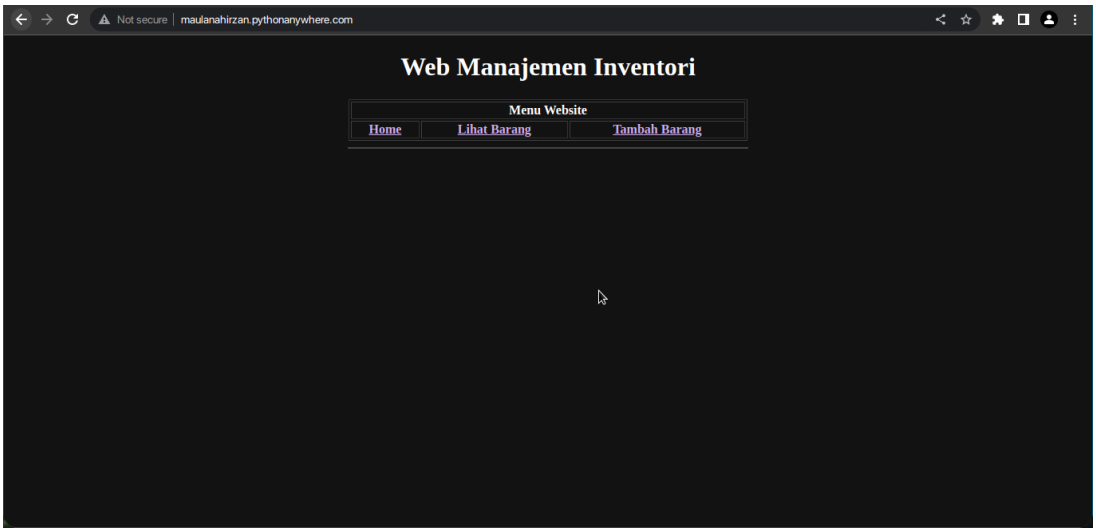
Gambar 2.23: WebApp : Mengunggah File index.html

24. Untuk mengetes **WebApp**, kembali ke **Tab WebApp** atau Buka **Tab Baru** menu **Web** di icon **Tiga Garis**. Klik **Reload xxx.pythonanywhere.com**.



Gambar 2.24: WebApp : Reload WebApp

25. Jika sudah reload, buka link website tersebut. Jika dilakukan dengan benar, maka website akan ditampilkan



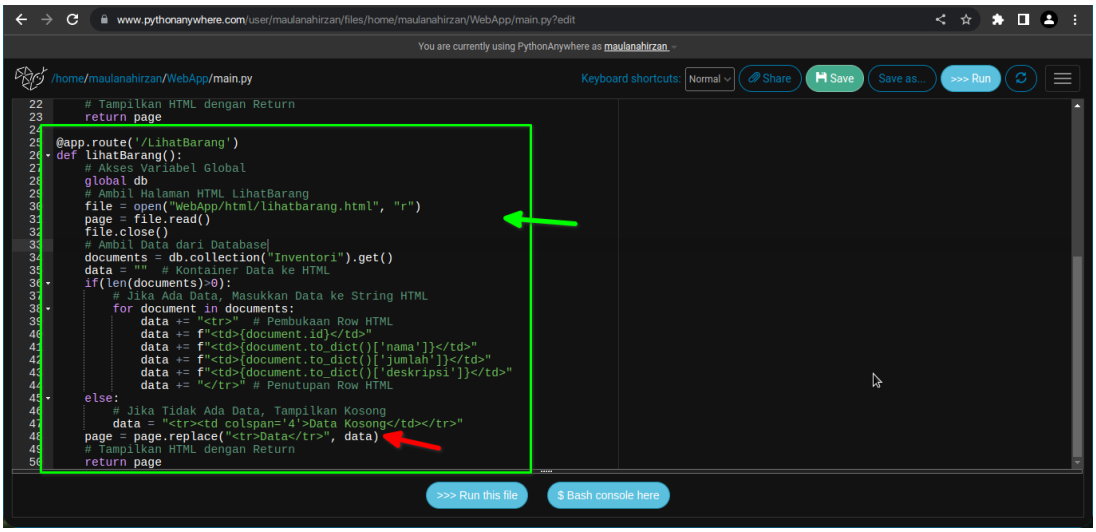
Gambar 2.25: WebApp : Halaman Awal WebApp

2.2.2 WebApp #2 - Lihat Barang

1. Untuk membuat halaman **Lihat Barang**, cukup membuka file **main.py** dan tambahkan kode rute baru seperti berikut:

Potongan Kode

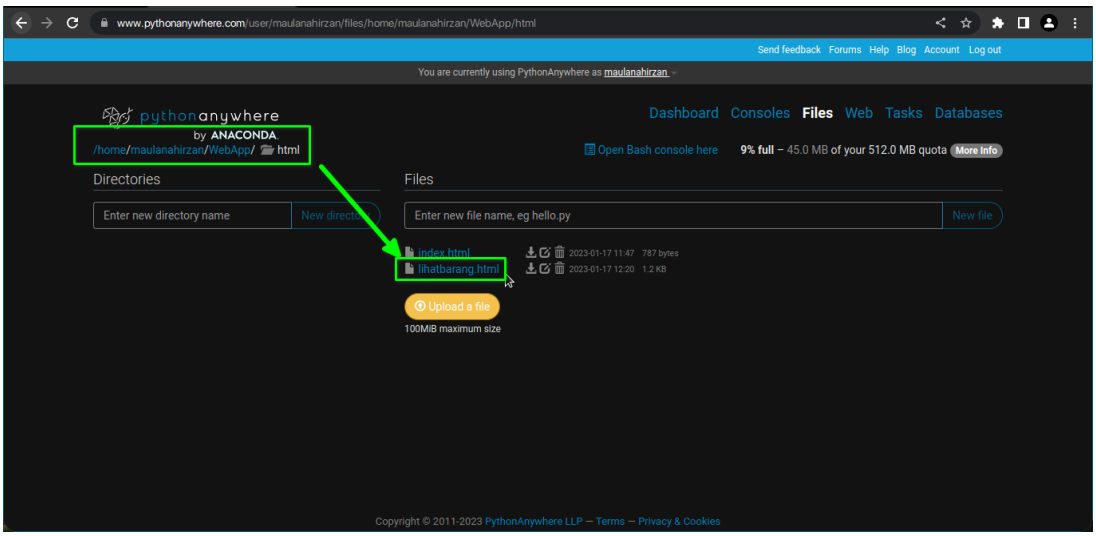
```
@app.route('/LihatBarang')
def lihatBarang():
    # Akses Variabel Global
    global db
    # Ambil Halaman HTML LihatBarang
    file = open("WebApp/html/lihatbarang.html", "r")
    page = file.read()
    file.close()
    # Ambil Data dari Database
    documents = db.collection("Inventori").get()
    data = ""  # Kontainer Data ke HTML
    if(len(documents)>0):
        # Jika Ada Data, Masukkan Data ke String HTML
        for document in documents:
            data += "<tr>"  # Pembukaan Row HTML
            data += f"<td>{document.id}</td>"
            data += f"<td>{document.to_dict()['nama']}</td>"
            data += f"<td>{document.to_dict()['jumlah']}</td>"
            data += f"<td>{document.to_dict()['deskripsi']}</td>"
            data += "</tr>" # Penutupan Row HTML
    else:
        # Jika Tidak Ada Data, Tampilkan Kosong
        data = "<tr><td colspan='4'>Data Kosong</td></tr>"
    page = page.replace("<tr>Data</tr>", data)
    # Tampilkan HTML dengan Return
    return page
```



Gambar 2.26: WebApp : Kode Lihat Barang WebApp

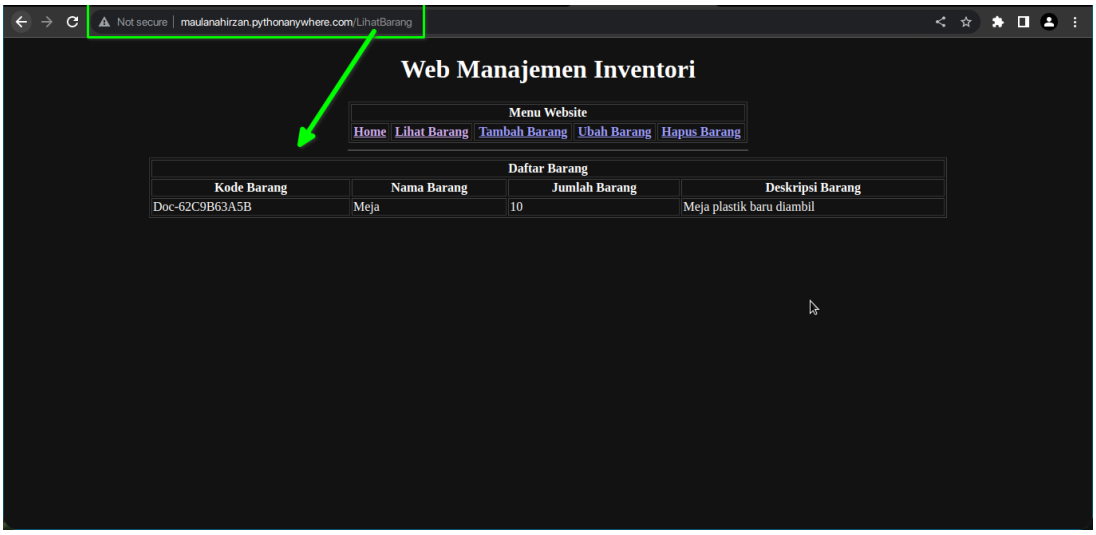
2. Jika diperhatikan di gambar, terdapat panah berwarna merah. Panah itu menunjukkan ke penggantian **Bagian Kode HTML** dengan teks <**tr**>**Data**</**tr**> dengan **Data Asli** hasil kueri.

3. Berikutnya adalah mengunggah file HTML **lihatbarang.html** ke folder **html**



Gambar 2.27: WebApp : Mengunggah File lihatbarang.html

4. Klik **Save** file **main.py**. Lalu **Reload WebApp** agar perubahan bisa dilihat. Buka halaman **Lihat Barang** dengan hasil seperti berikut



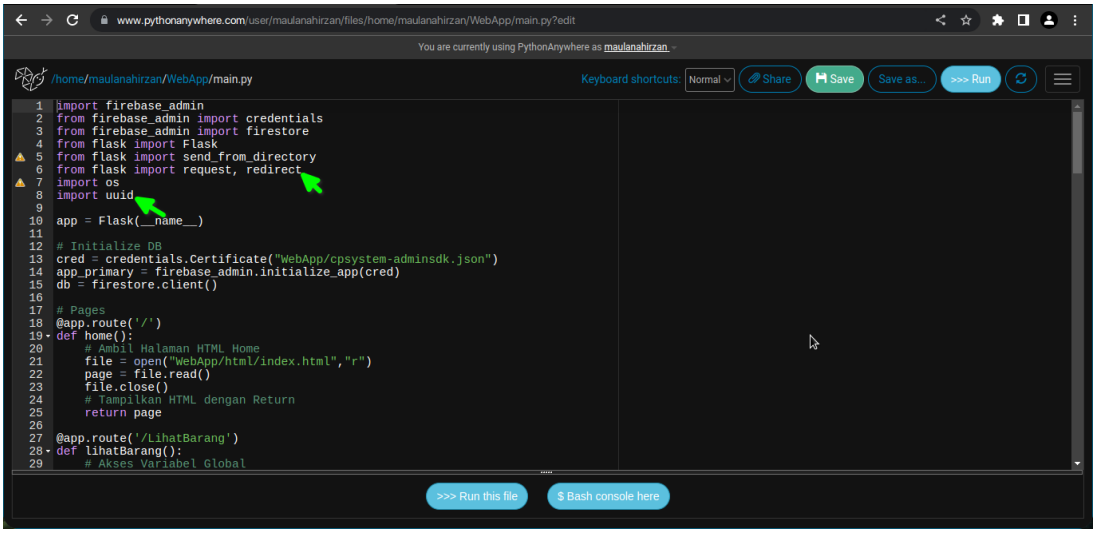
Gambar 2.28: WebApp : Tampilan Lihat Barang

2.2.3 WebApp #3 - Tambah Barang

1. Langkah berikutnya dalam WebApp adalah membuat menu **Tambah Barang**
2. Buka kembali file **main.py** lalu tambahkan kode berikut

Potongan Kode

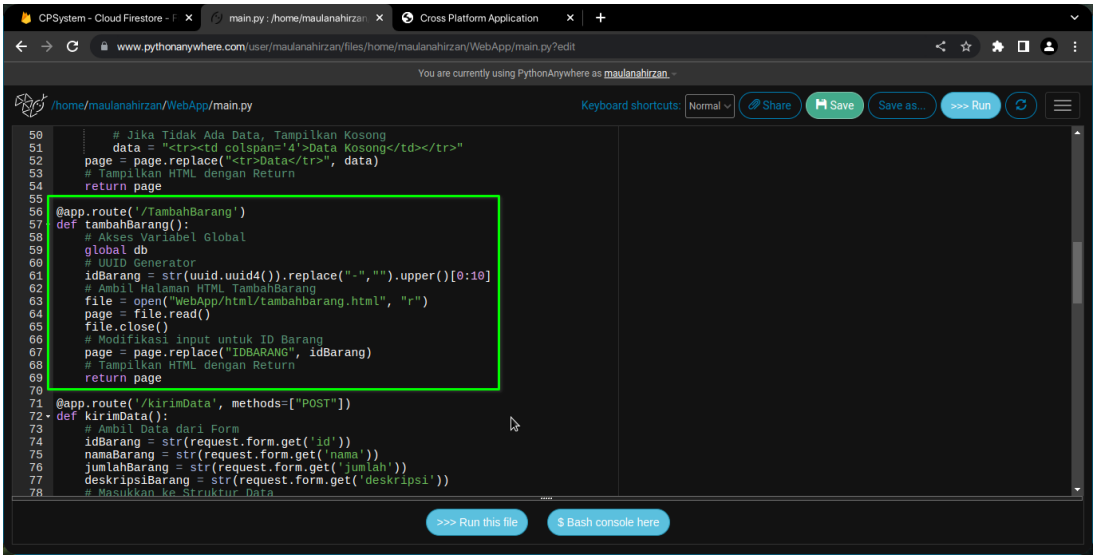
```
from flask import request, redirect
import uuid
```



Gambar 2.29: WebApp : Kode Impor Tambahan

Potongan Kode

```
@app.route('/TambahBarang')
def tambahBarang():
    # Akses Variabel Global
    global db
    # UUID Generator -> Kapital -> Potong 10 Karakter
    idBarang = str(uuid.uuid4()).replace("-","").upper()[0:10]
    # Ambil Halaman HTML TambahBarang
    file = open("WebApp/html/tambahbarang.html", "r")
    page = file.read()
    file.close()
    # Modifikasi input untuk ID Barang di HTML
    page = page.replace("IDBARANG", idBarang)
    # Tampilkan HTML dengan Return
    return page
```

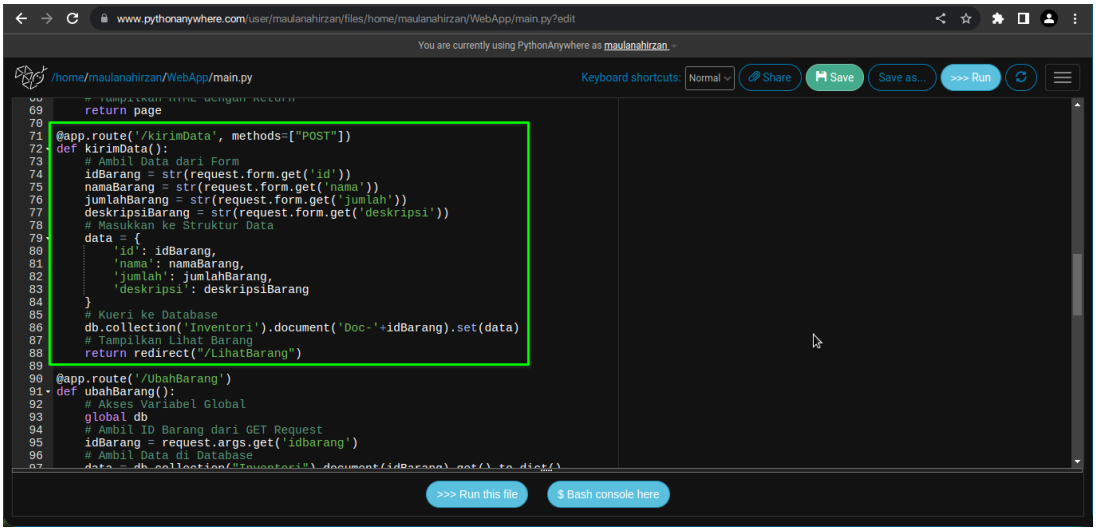


Gambar 2.30: WebApp : Kode Menampilkan Halaman Tambah Data

3. Kode tersebut hanya akan menampilkan halaman **Tambah Barang** namun tidak bisa melakukan kueri ke database. Tambahkan kode berikut untuk proses kueri ke database.

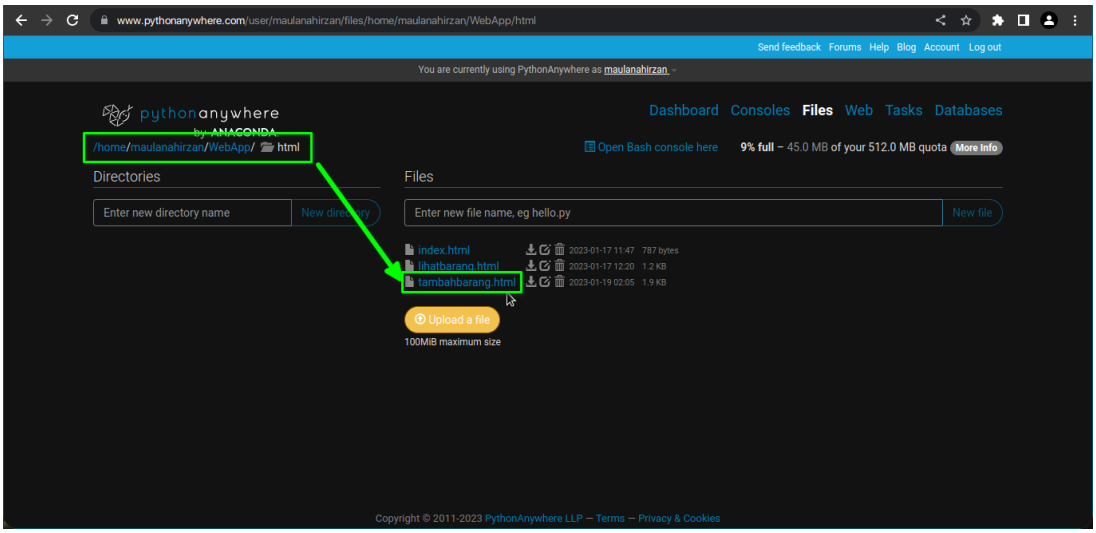
Potongan Kode

```
@app.route('/kirimData', methods=["POST"])
def kirimData():
    # Ambil Data dari Form
    idBarang = str(request.form.get('id'))
    namaBarang = str(request.form.get('nama'))
    jumlahBarang = str(request.form.get('jumlah'))
    deskripsiBarang = str(request.form.get('deskripsi'))
    # Masukkan ke Struktur Data
    data = {
        'id': idBarang,
        'nama': namaBarang,
        'jumlah': jumlahBarang,
        'deskripsi': deskripsiBarang
    }
    # Kueri ke Database
    db.collection('Inventori').document('Doc-'+idBarang).set(data)
    # Tampilkan Lihat Barang
    return redirect("/LihatBarang")
```



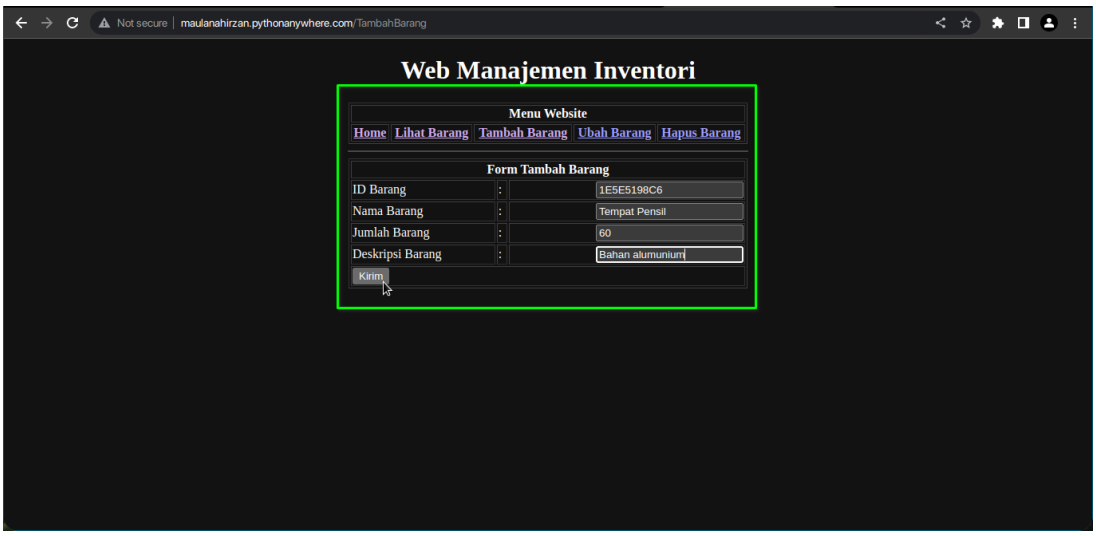
Gambar 2.31: WebApp : Kode Kirim Data

4. Berikutnya unggah file **tambahbarang.html** ke folder **html**

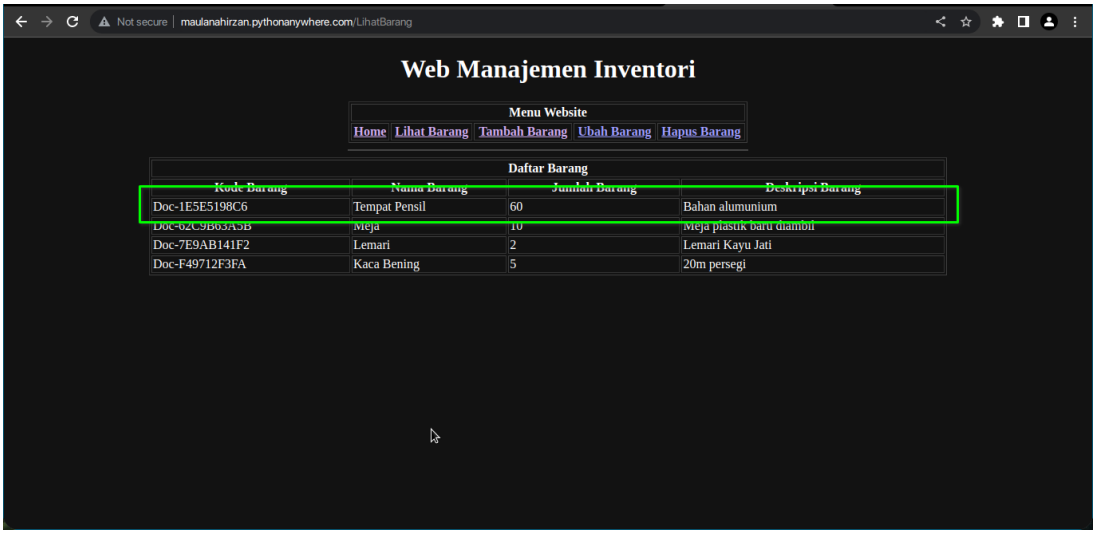


Gambar 2.32: WebApp : Unggah File tambahdata.html

5. Reload WebApp lalu coba tambahkan data barang baru



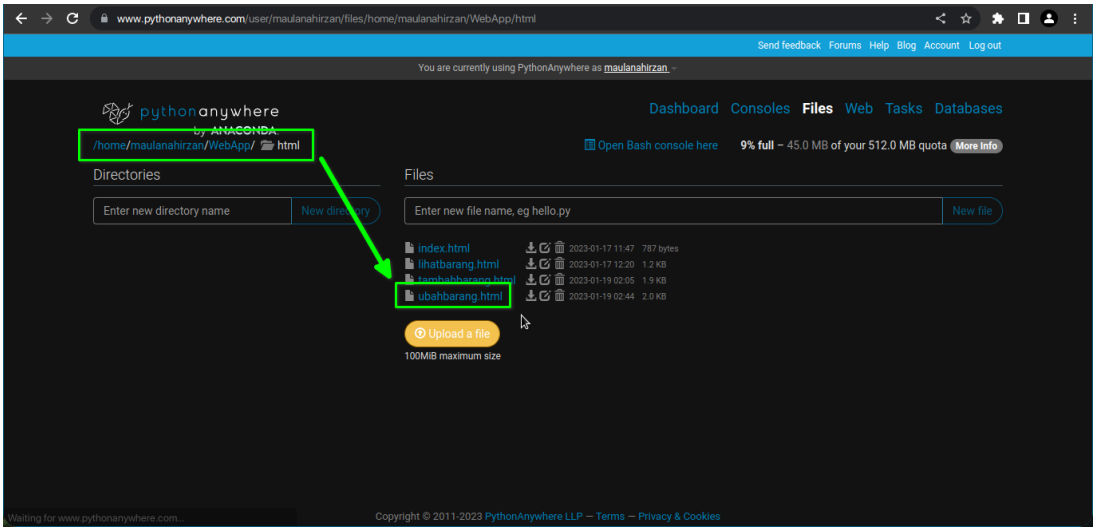
Gambar 2.33: WebApp : Uji Coba Tambah Data #1



Gambar 2.34: WebApp : Uji Coba Tambah Data #2

2.2.4 WebApp #4 - Ubah Barang

1. Untuk merubah barang yang ada di database, unggah file **ubahbarang.html** ke folder **html**



Gambar 2.35: WebApp : Mengunggah File ubahbarang.html

2. Agar bisa melakukan perubahan dari halaman **Lihat Barang**. Ubah kode di **main.py** berikut dari

Potongan Kode

```
data += f"<td>{document.id}</td>"
```

Gambar 2.36: WebApp : Kode Sebelum Diubah

Potongan Kode

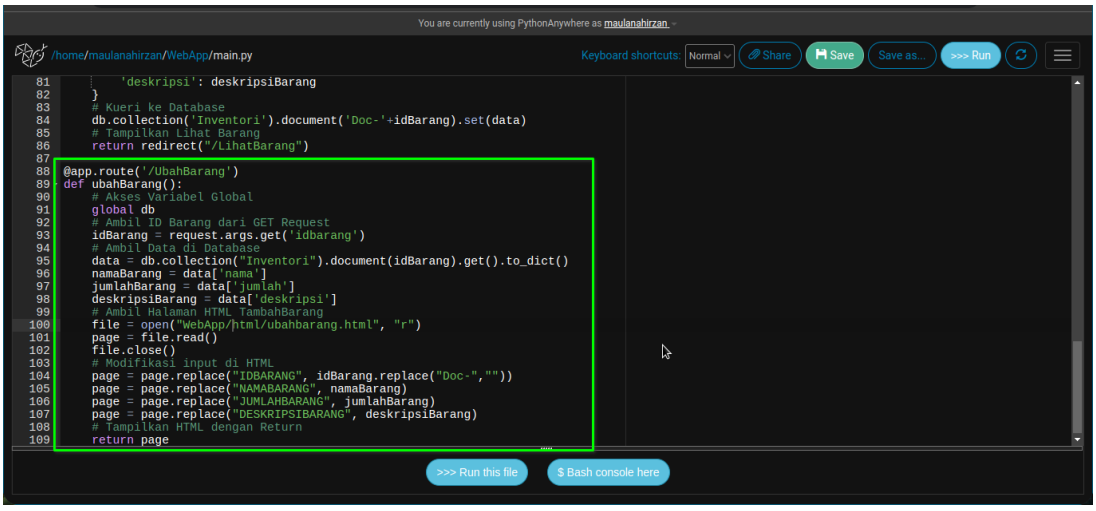
```
data += f"<td>{document.id} - " \
        f"<a href='/UbahBarang?idbarang={document.id}'>Ubah</a></td>"
```

Gambar 2.37: WebApp : Kode Setelah Diubah

3. Lalu buka kembali file **main.py** dan tambahkan kode berikut

Potongan Kode

```
@app.route('/UbahBarang')
def ubahBarang():
    # Akses Variabel Global
    global db
    # Ambil ID Barang dari GET Request
    idBarang = request.args.get('idbarang')
    # Ambil Data di Database
    data = db.collection("Inventori").document(idBarang).get().to_dict()
    namaBarang = data['nama']
    jumlahBarang = data['jumlah']
    deskripsiBarang = data['deskripsi']
    # Ambil Halaman HTML TambahBarang
    file = open("WebApp/html/ubahbarang.html", "r")
    page = file.read()
    file.close()
    # Modifikasi input di HTML
    page = page.replace("IDBARANG", idBarang.replace("Doc-",""))
    page = page.replace("NAMABARANG", namaBarang)
    page = page.replace("JUMLAHBARANG", jumlahBarang)
    page = page.replace("DESKRIPSIBARANG", deskripsiBarang)
    # Tampilkan HTML dengan Return
    return page
```

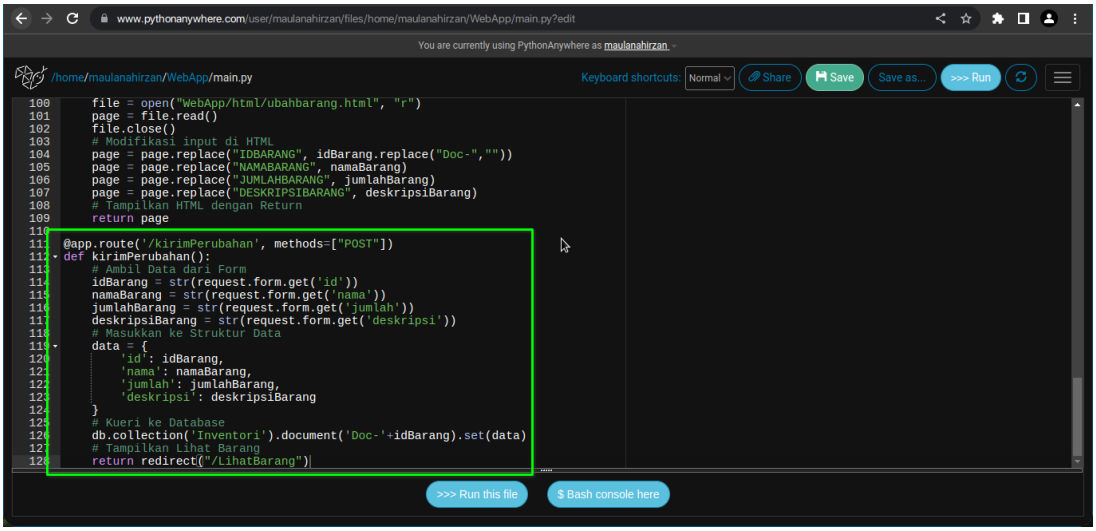


Gambar 2.38: WebApp : Kode Menampilkan Ubah Data

4. Agar bisa dikirimkan pembaruannya, masukkan kode berikut

Potongan Kode

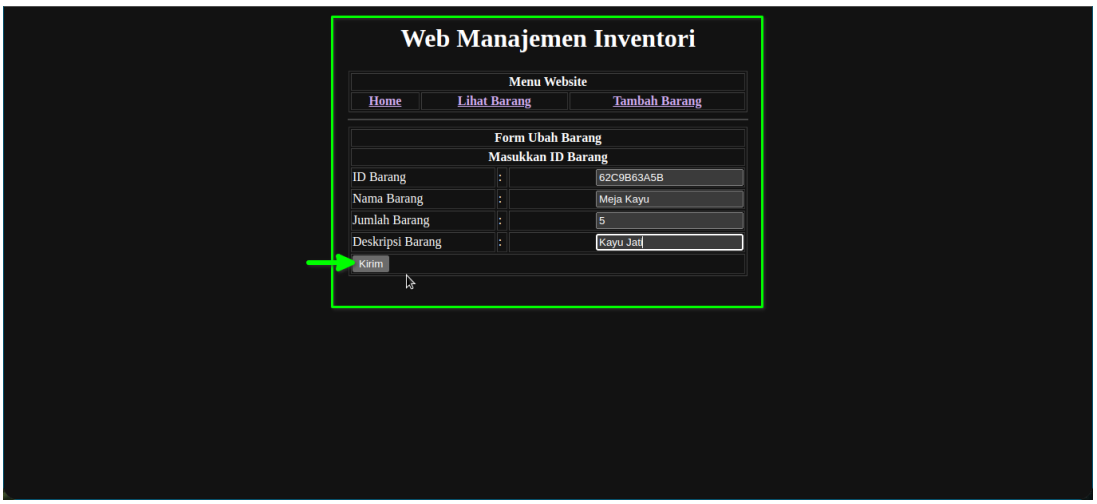
```
@app.route('/kirimPerubahan', methods=["POST"])
def kirimPerubahan():
    # Ambil Data dari Form
    idBarang = str(request.form.get('id'))
    namaBarang = str(request.form.get('nama'))
    jumlahBarang = str(request.form.get('jumlah'))
    deskripsiBarang = str(request.form.get('deskripsi'))
    # Masukkan ke Struktur Data
    data = {
        'id': idBarang,
        'nama': namaBarang,
        'jumlah': jumlahBarang,
        'deskripsi': deskripsiBarang
   }
    # Kueri ke Database
    db.collection('Inventori').document('Doc-'+idBarang).set(data)
    # Tampilkan Lihat Barang
    return redirect("/LihatBarang")
```



Gambar 2.39: WebApp : Kode Pengirim Perubahan Data

5. Reload Web App lalu coba ubah barang

Gambar 2.40: WebApp : Uji Coba Ubah Data #1

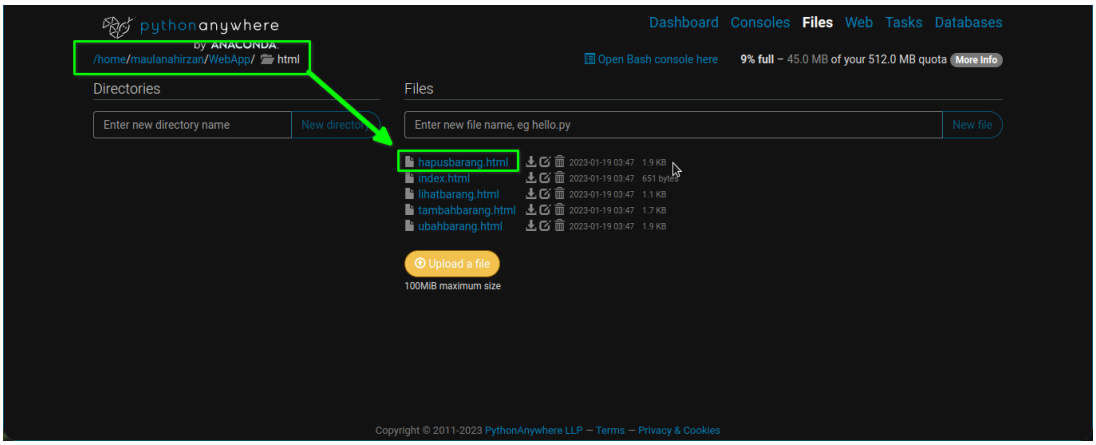


Gambar 2.41: WebApp : Uji Coba Ubah Data #2

Gambar 2.42: WebApp : Uji Coba Ubah Data #3

2.2.5 WebApp #5 - Hapus Barang

1. Sebelum masuk ke tahap koding, unggah file **hapusbarang.html** ke folder **html**

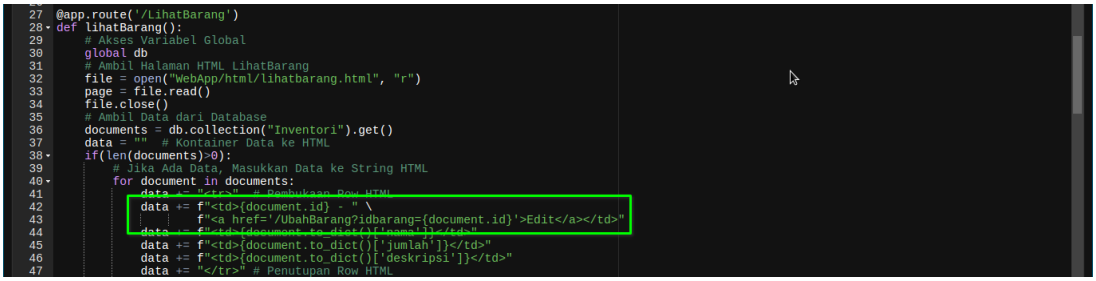


Gambar 2.43: WebApp : Menunggah File hapusbarang.html

2. Ubah kode **Lihat Barang**, dan tambahkan kode seperti berikut

Potongan Kode

```
data += f"<td>{document.id} - " \
        f"<a href='/UbahBarang?idbarang={document.id}'>Ubah</a></td>"
```



Gambar 2.44: WebApp : Kode Sebelum Diubah

Potongan Kode

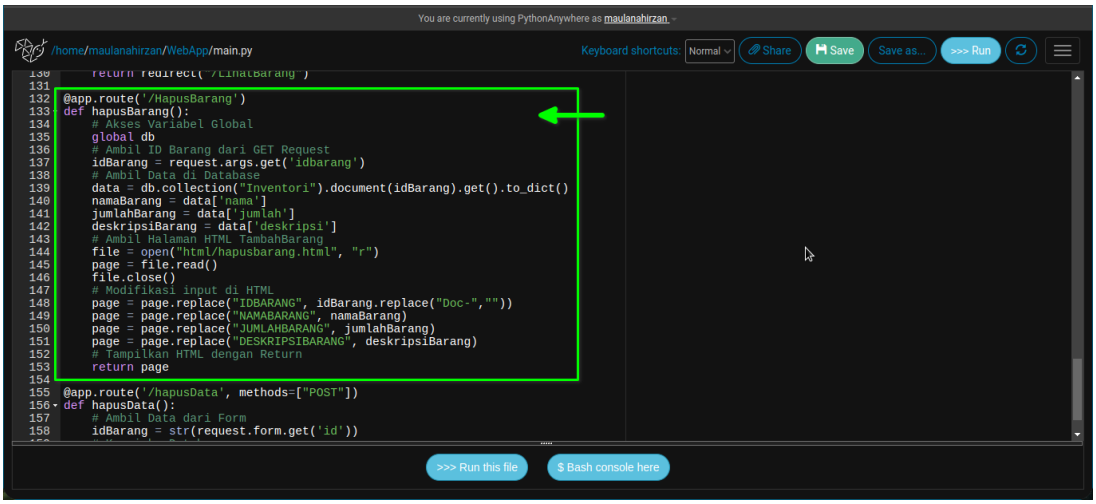
```
data += f"<td>{document.id} - " \
        f"<a href='/UbahBarang?idbarang={document.id}'>Ubah</a> - " \
        f"<a href='/HapusBarang?idbarang={document.id}'>Hapus</a></td>"
```

Gambar 2.45: WebApp : Kode Sesudah Diubah

3. Lalu masukkan kode berikut ke **main.py**

Potongan Kode

```
@app.route('/HapusBarang')
def hapusBarang():
    # Akses Variabel Global
    global db
    # Ambil ID Barang dari GET Request
    idBarang = request.args.get('idbarang')
    # Ambil Data di Database
    data = db.collection("Inventori").document(idBarang).get().to_dict()
    namaBarang = data['nama']
    jumlahBarang = data['jumlah']
    deskripsiBarang = data['deskripsi']
    # Ambil Halaman HTML TambahBarang
    file = open("WebApp/html/hapusbarang.html", "r")
    page = file.read()
    file.close()
    # Modifikasi input di HTML
    page = page.replace("IDBARANG", idBarang.replace("Doc-",""))
    page = page.replace("NAMABARANG", namaBarang)
    page = page.replace("JUMLAHBARANG", jumlahBarang)
    page = page.replace("DESKRIPSIBARANG", deskripsiBarang)
    # Tampilkan HTML dengan Return
    return page
```

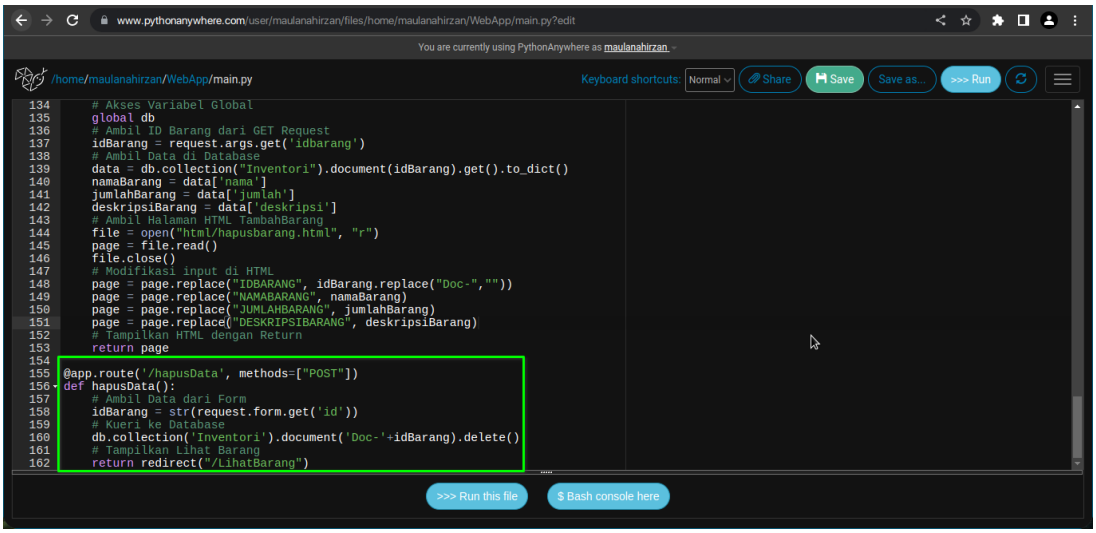


Gambar 2.46: WebApp : Kode Menampilkan Hapus Barang

4. Untuk eksekusi penghapusan, masukkan kode berikut:

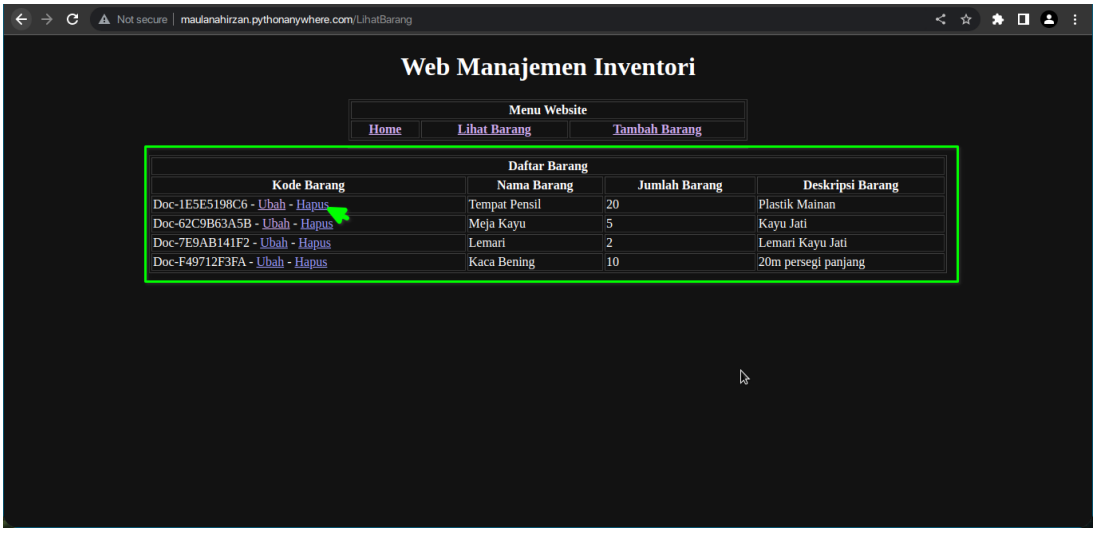
Potongan Kode

```
@app.route('/hapusData', methods=["POST"])
def hapusData():
    # Ambil Data dari Form
    idBarang = str(request.form.get('id'))
    # Kueri ke Database
    db.collection('Inventori').document('Doc-'+idBarang).delete()
    # Tampilkan Lihat Barang
    return redirect("/LihatBarang")
```

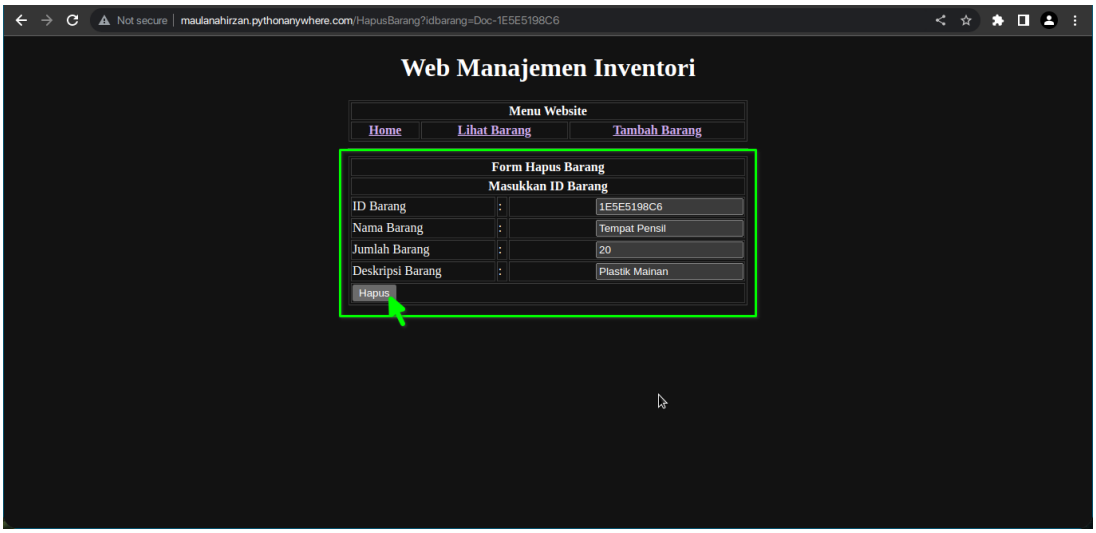


Gambar 2.47: WebApp : Kode Penghapus Barang

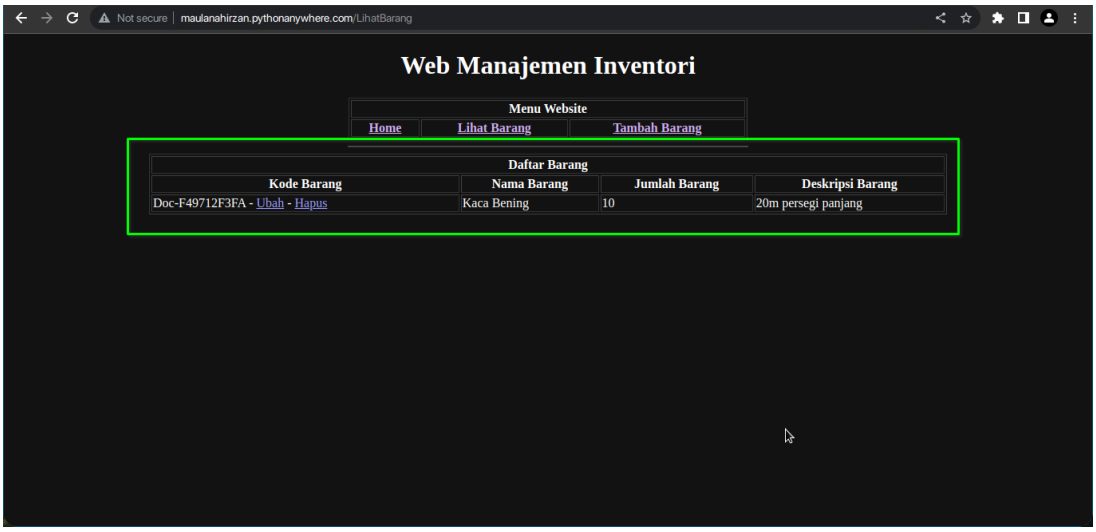
5. Reload **WebApp**, dan coba untuk menghapus item



Gambar 2.48: WebApp : Uji Coba Hapus Data #1



Gambar 2.49: WebApp : Uji Coba Hapus Data #2



Gambar 2.50: WebApp : Uji Coba Hapus Data #3

6. Selesai