**Modul Parktikum**

**Pemrograman Deklaratif**

1. **Mengenal Apa itu firebase**

Firebase adalah suatu layanan dari Google untuk memberikan kemudahan bahkan mempermudah para developer aplikasi dalam mengembangkan aplikasinya. 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*.

Singkat cerita mengenai sejarah dari Firebase didirikan pertama kali pada tahun 2011 oleh Andrew Lee dan James Tamplin. Produk Firebase yang pertama kali adalah Realtime Database. Realtime Database digunakan developer untuk menyimpan data dan *synchronize* ke banyak *user*. Kemudian ia berkembang sebagai layanan pengembang aplikasi. Pada bulan Oktober 2014, perusahaan tersebut diakuisisi oleh Google.

Mengenai segi layanan, dulu Firebase memberikan *service trial*(percobaan), namun saat ini kamu bisa memanfaatkan dan menggunakan layanan Firebase secara *free* (gratis). Tentu saja dengan adanya batasan-batasan tertentu.

Layanan-layanan yang tersedia dari Firebase ada 2 pilihan, di antaranya:

* **SPARK**: kita bisa menggunakan layanan secara gratis.
* **BLAZE** : kita akan dikenakan biaya sesuai dengan pemakaian layanan.

1. **Mengetahui Jenis atau Fitur Firebase**

Selanjutnya kita akan membahas jenis atau fitur dari Firebase itu sendiri, berikut detailnya:

1. Firebase Analytics

Fitur *Analytics*adalah salah satu fitur pada Firebase yang digunakan sebagai koleksi data dan *reporting*untuk aplikasi Android maupun iOS. Koleksi data pun bervariasi. Sebagai contoh, kamu dapat membuat suatu laporan atau *report*untuk pengguna aplikasi di negara Indonesia saja, atau mungkin negara lain seperti Singapura. Kamu juga bisa melihat bagian mana saja dari aplikasi yang paling sering digunakan oleh *user*.

Fitur ini mempunyai kelebihan yang memungkinkan kita untuk bisa membuat segmentasi *user*berdasarkan *user attribute. User attribute*adalah suatu parameter yang bisa kita gunakan sebagai *filter* yangbertujuan untuk *reporting*dan notifikasi. Contohnya pada aplikasi *online shop*. Dengan *user attribute,*kamu bisa tahu jumlah *user*yang membeli *handphone*merk ‘O’ atau bahkan bisa mencari tahu jam berapa transaksi yang dilakukan *user*sering terjadi.

1. Firebase Cloud Messaging and Notifications

FCM (*Firebase Cloud Messaging)*yaitu menyediakan koneksi yang handal dan tentunya hemat baterai antar server maupun antar *device*. Sehingga kamu dapat mengirim dan menerima pesan serta notifikasi di Android, iOS, dan web tanpa perlu biaya.

Untuk menargetkan pesan lanjutan, kamu bisa targetkan pesan dengan mudah menggunakan segment yang telah ditentukan sebelumnya yakni menggunakan demografi dan behavior/perilaku. Anda dapat menargetkan pesan ke perangkat yang telah berlangganan pada topik tertentu. Selain itu, Anda bisa juga menargetkan hanya ke satu perangkat untuk mendapatkan informasi data yang terperinci. Biasanya ini dilakukan untuk proses pengujian.

Pesan notifikasi ini terintegrasi sepenuhnya dengan Google Analytics for Firebase, sehingga kamu memiliki akses pada interaksi dan tracking konversi secara detaill. Nah, Anda dapat memantau suatu efektivitas dari satu dashboard tanpa perlu *coding*atau membuat program sendiri.

1. Firebase Authentication

Firebase Authentication adalah salah satu layanan *back-end,*fitur Android dan iOS, SDK yang mudah digunakan, dan tampilan *interfaces*yang siap pakai untuk mengautentikasi pengguna ke aplikasi yang kamu buat. Firebase Authentication mendukung autentikasi menggunakan nomor telepon, sandi, penyedia identitas gabungan populer seperti seperti Google, Facebook, dan sebagainya.

Firebase Authentication terintegrasi dengan fitur layanan Firebase lainnya. Sistem ini memanfaatkan berbagai jenis standar industri, seperti OAuth 2.0 dan OpenID Connect, yang memudahkan integrasi dengan *backend*khusus buatanmu.

Kamu juga dapat memudahkan pengguna untuk *login*ke aplikasi dengan menggunakan fitur Firebase UI (tampilan *interfaces*), sebagai alternatif full drop-in authentication.

1. Firebase Cloud Firestore

Cloud Firestore adalah *database* yang bersifat fleksibel dan terukur untuk pengembangan perangkat seperti seluler, web, dan server di Firebase dan *Google Cloud Platform*. Seperti halnya Firebase Realtime Database, Cloud Firestore membuat datamu tetap terkoneksi di aplikasi *user*melalui *listener realtime* dan menawarkan layanan secara *offline*untuk aplikasi seluler dan web. Dengan begitu, kamu dapat membuat aplikasi yang *powerfull*, responsif, dan mampu bekerja tanpa bergantung pada latensi koneksi internet.

Cloud Firestore merupakan database NoSQL yang dihosting di cloud dan dapat diakses melalui SDK real oleh aplikasi iOS, Android dan web.

1. Firebase Realtime Database

Firebase Realtime *Database*adalah *database*yang di-host melalui cloud. Data disimpan dan dieksekusi dalam bentuk *JSON* dan disinkronkan secara *realtime*ke setiap *user*yang terkoneksi. Hal ini berfungsi memudahkan kamu dalam mengelola suatu *database*dengan skala yang cukup besar. Ketika kamu membuat aplikasi lintas-platform/multiplatform menggunakan SDK Android, iOS, dan juga JS (*JavaScript),* semua pengguna akan berbagi sebuah instance Realtime *Database*dan menerima *update*-an data secara serentak dan otomatis.

Kemampuan lain dari Firebase Realtime *Database*adalah tetap responsif bahkan saat *offline*karena SDK Firebase Realtime *Database* menyimpan data langsung ke *disk device*atau memori lokal. Setelah perangkat terhubung kembali dengan internet, perangkat pengguna (*user*) akan menerima setiap perubahan yang terjadi.

1. Firebase Hosting

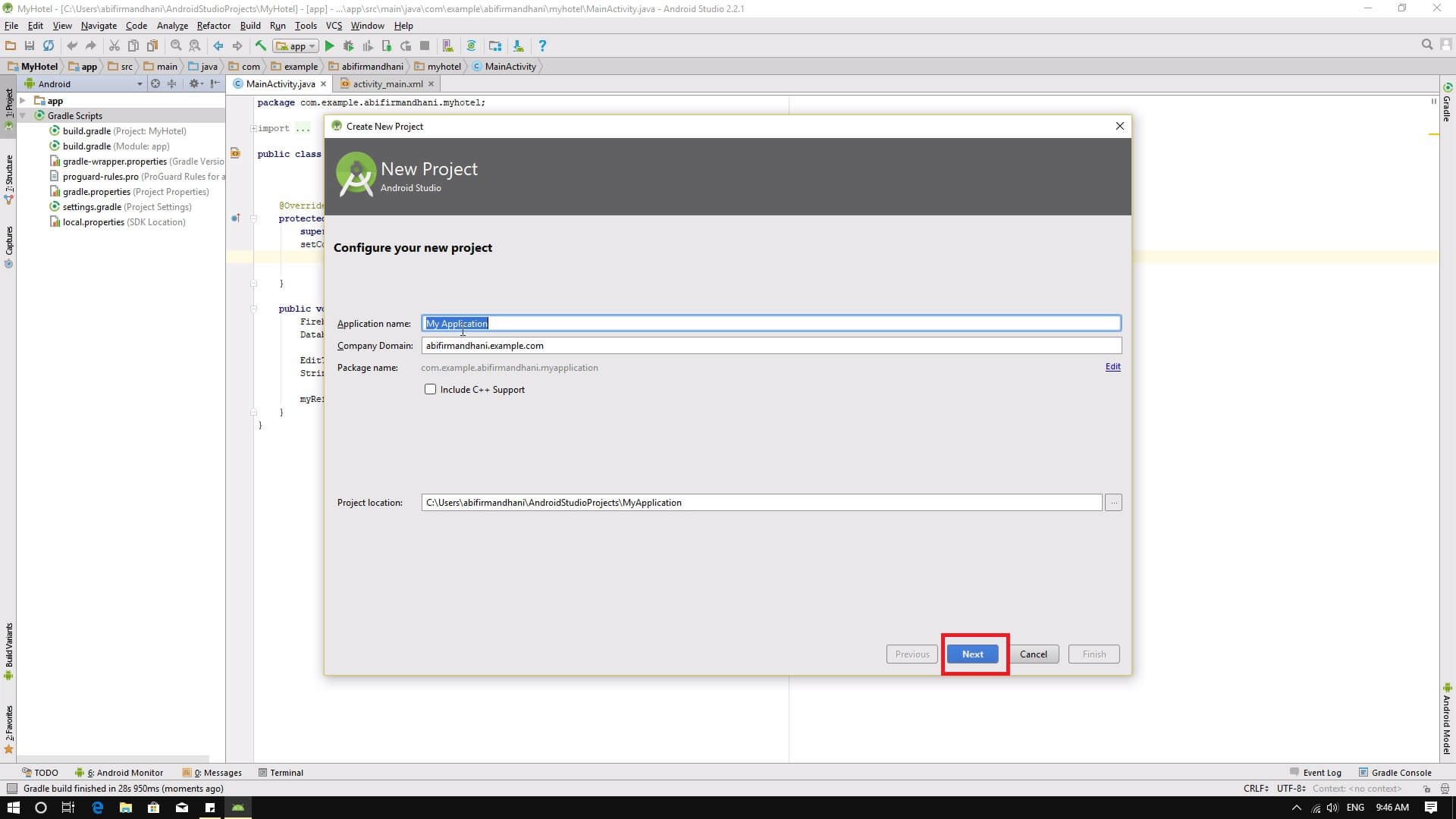
Selanjutnya ada Firebase Hosting, suatu layanan *hosting*konten web. Hanya dengan satu instruksi, kamu dapat mengimplementasikan aplikasi web serta menyajikan konten statis maupun dinamis ke CDN  (jaringan penayangan konten) global dengan cepat.

Kegunaan dari Firebase Hosting itu sendiri yaitu mampu menayangkan konten melalui koneksi yang begitu aman, mengirimkan konten secara cepat, dan mendukung semua jenis konten untuk di *hosting*, mulai dari file HTML dan CSS  hingga API atau layanan mikro Express.js.

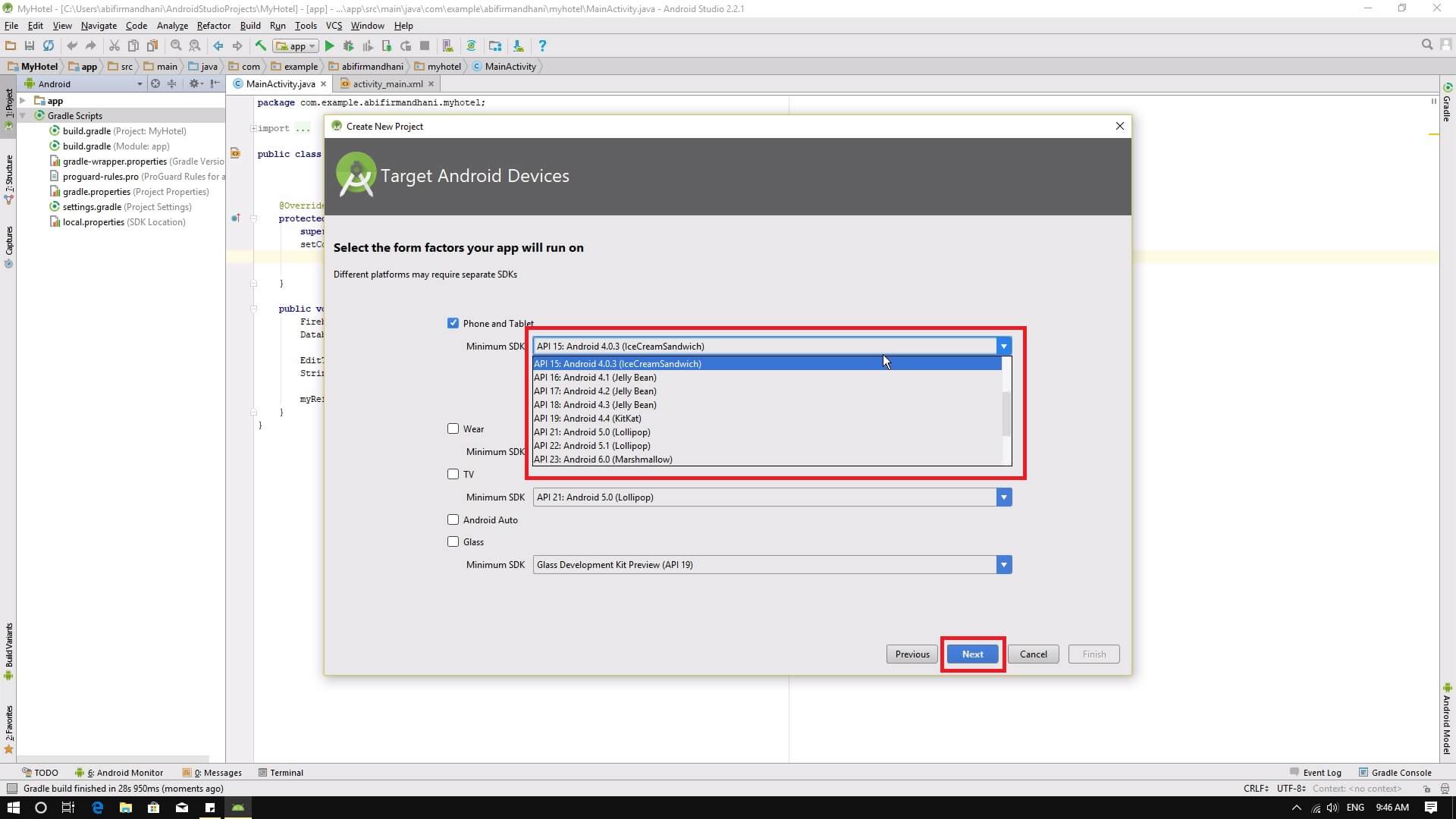
1. **Integrasi Firebase dengan Aplikasi Android**

Adapun langkah-langkah dalam melakukan integrase dari firebase Databases dengan aplikasi android yang dibuat menggunakan tools android studio sebagai berikut:

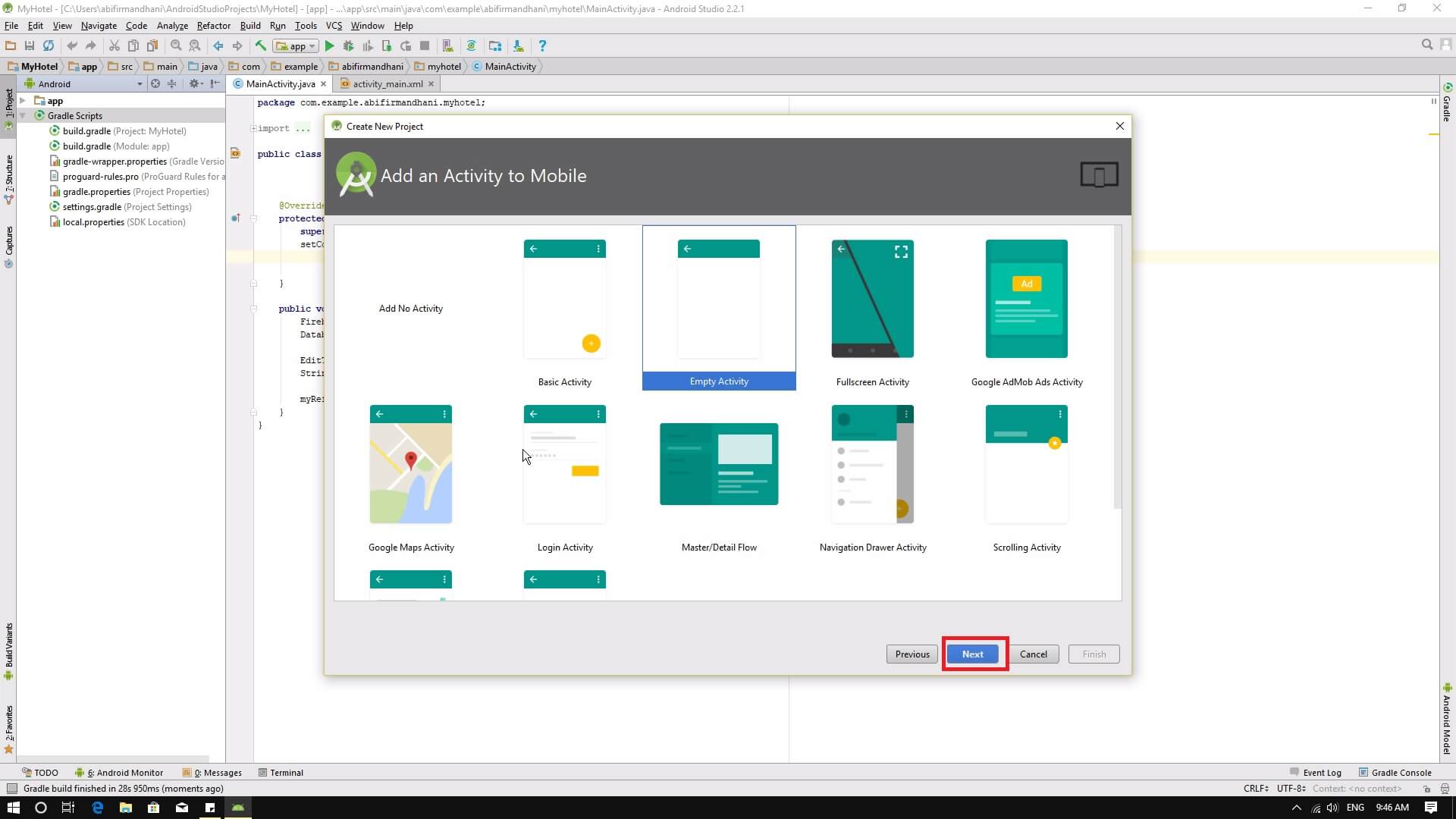
* 1. **Buat projek baru** di Android Studio kamu. Buat nama pada projek kamu dan klik tombol***next.***



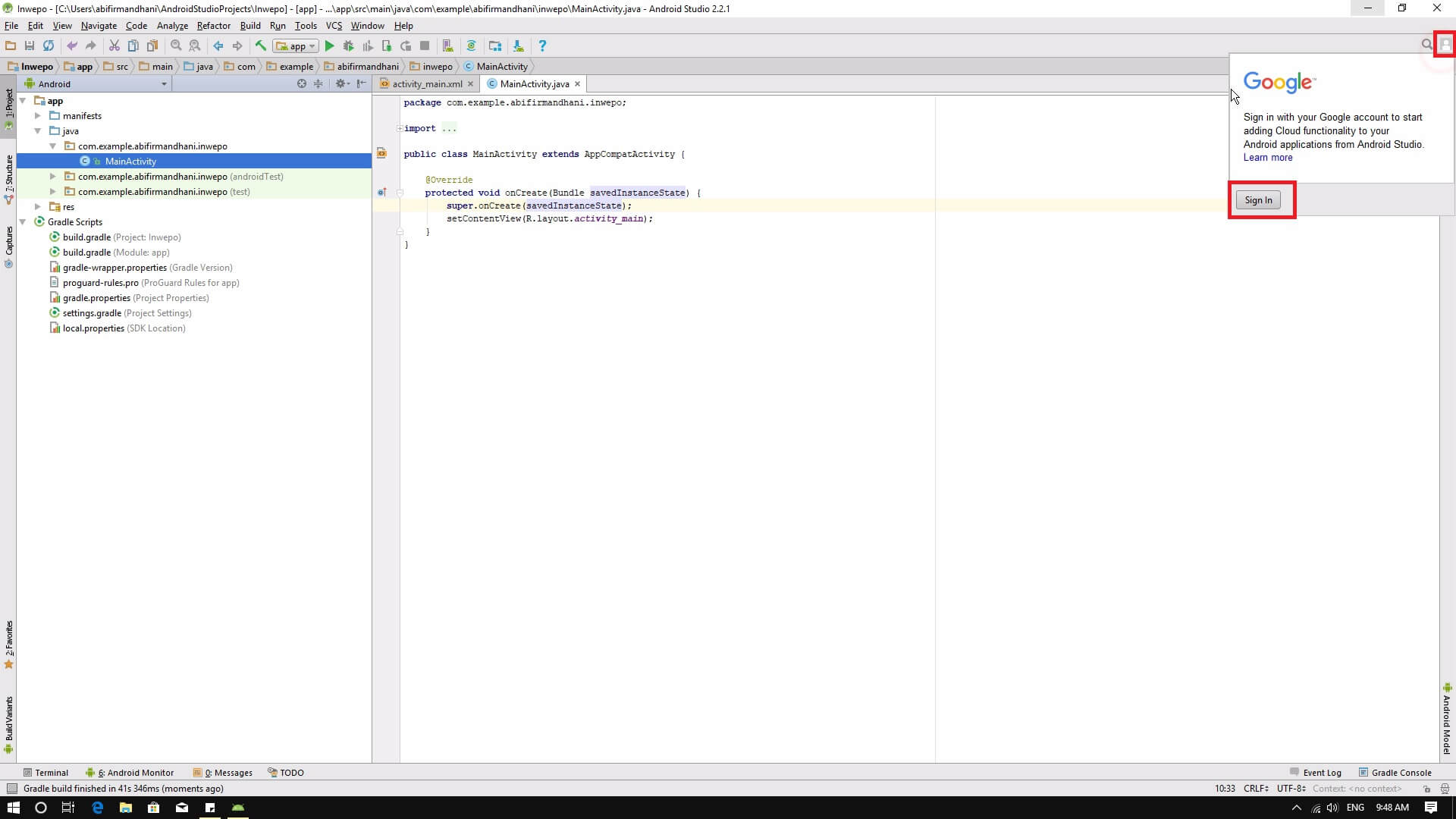
2. **Pilih *platform***yang akan kamu kembangkan dan **pilih *API*** nya juga, lalu klik tombol***next.***



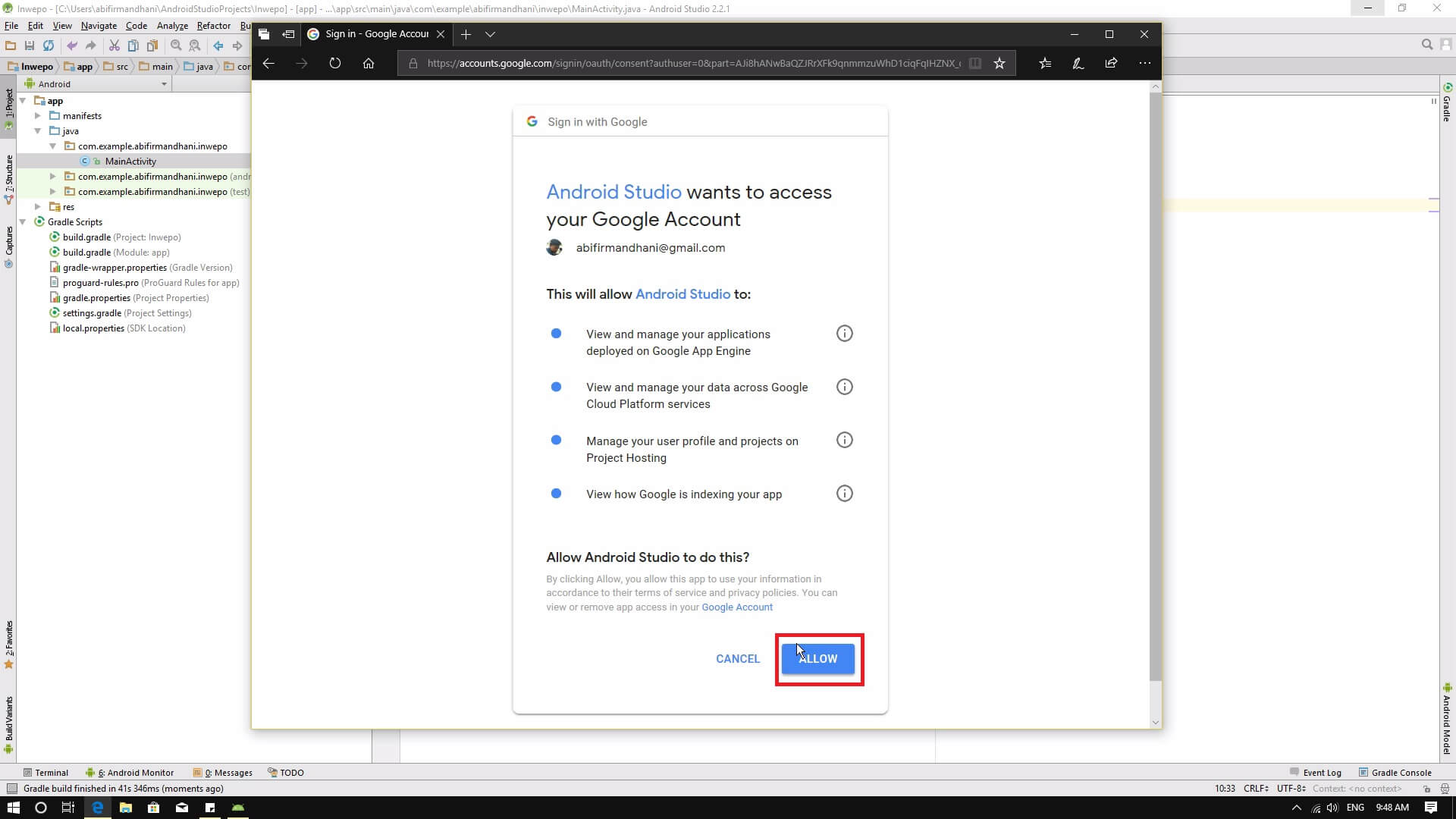
3. **Pilih *activity*** awal yang akan kamu gunakan. Pada tutorial kali ini akan menggunakan *activity* kosong.



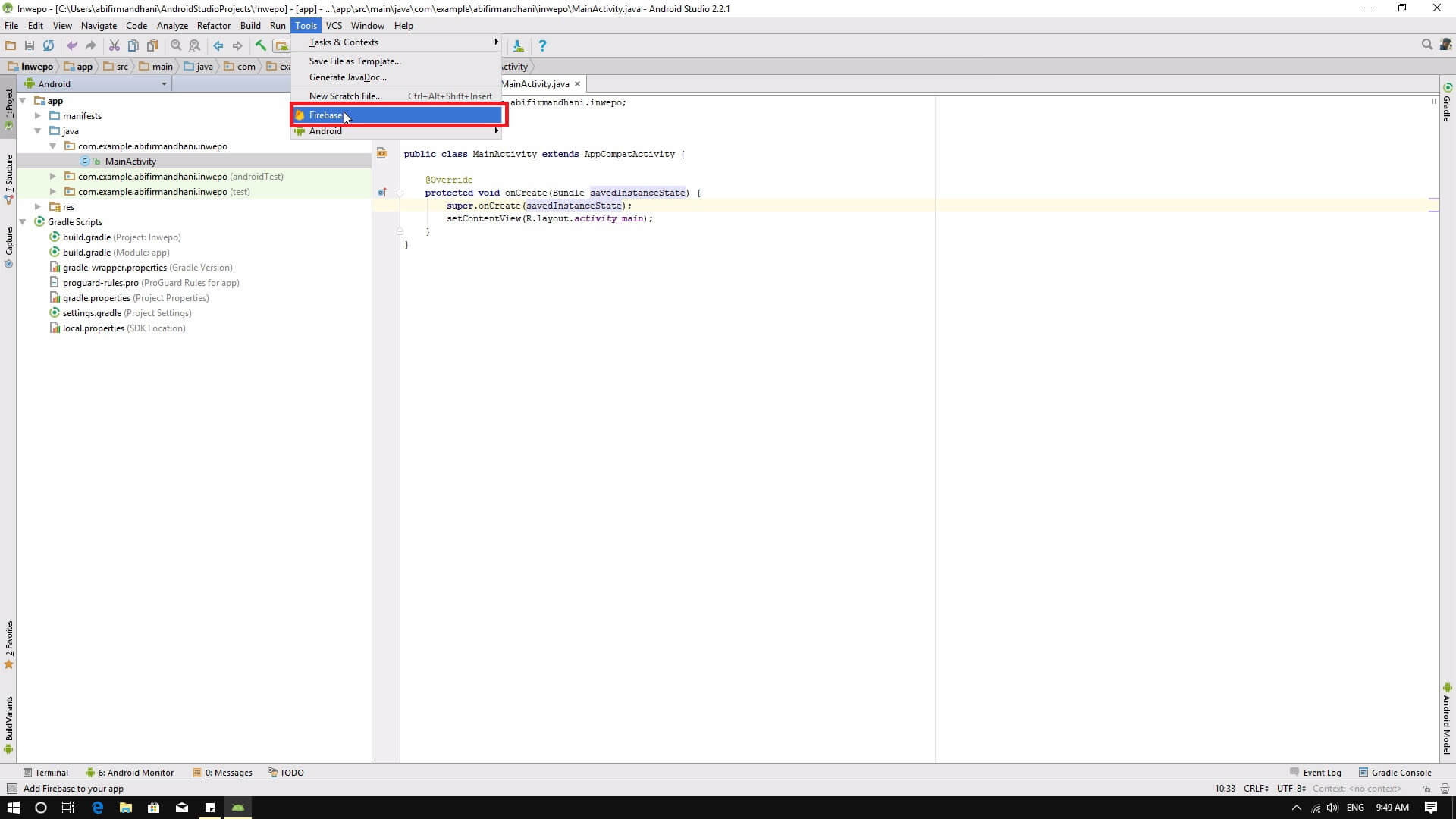
4. Saat sudah masuk ke dalam projek kamu, klik ikon user di pojok kanan atas dan ***Sign in* dengan akun Google** kamu



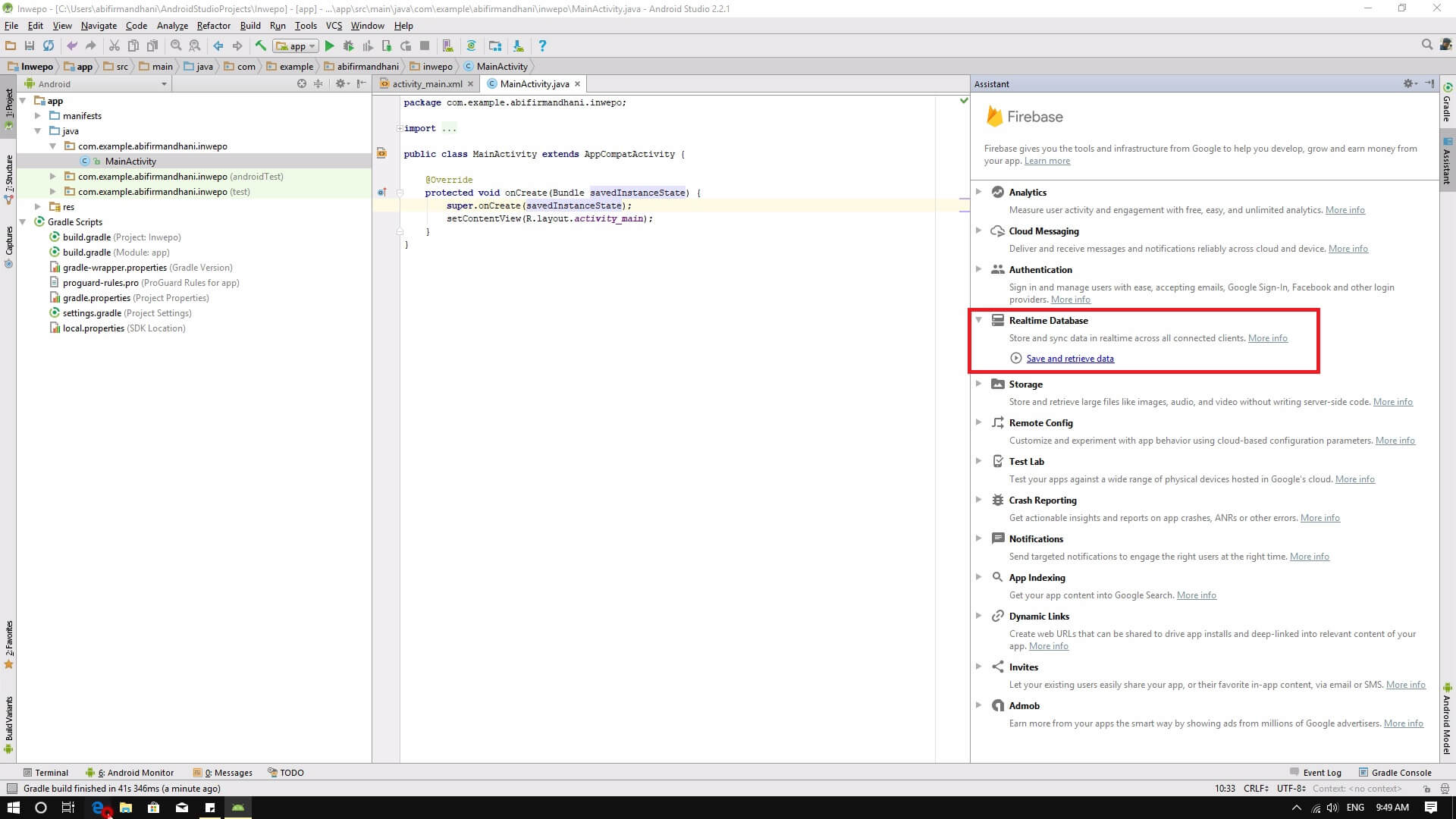
5. Klik tombol***allow*** untuk melanjutkan.



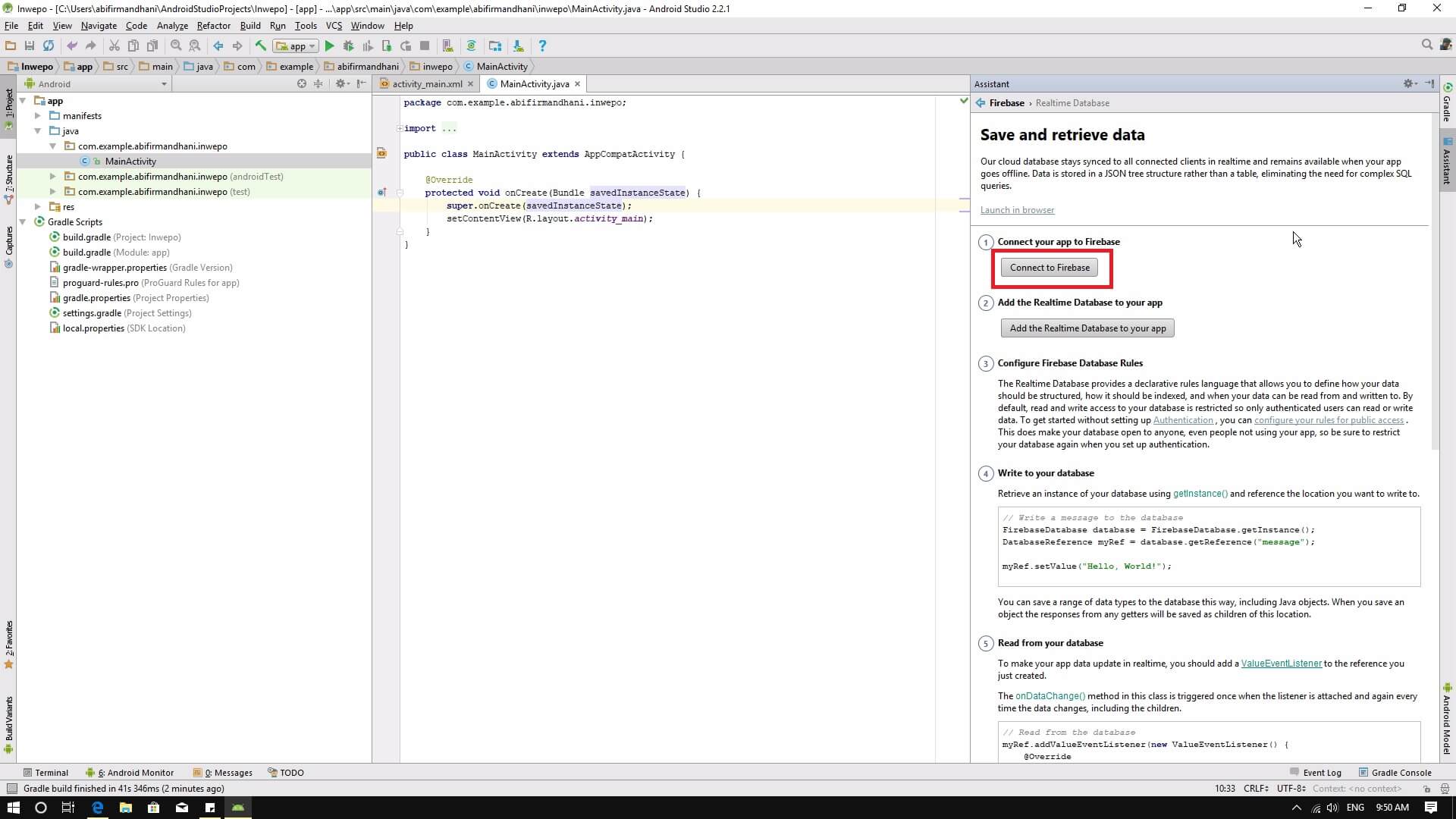
6. Lalu klik menu***tools***dan pilih**Firebase.**



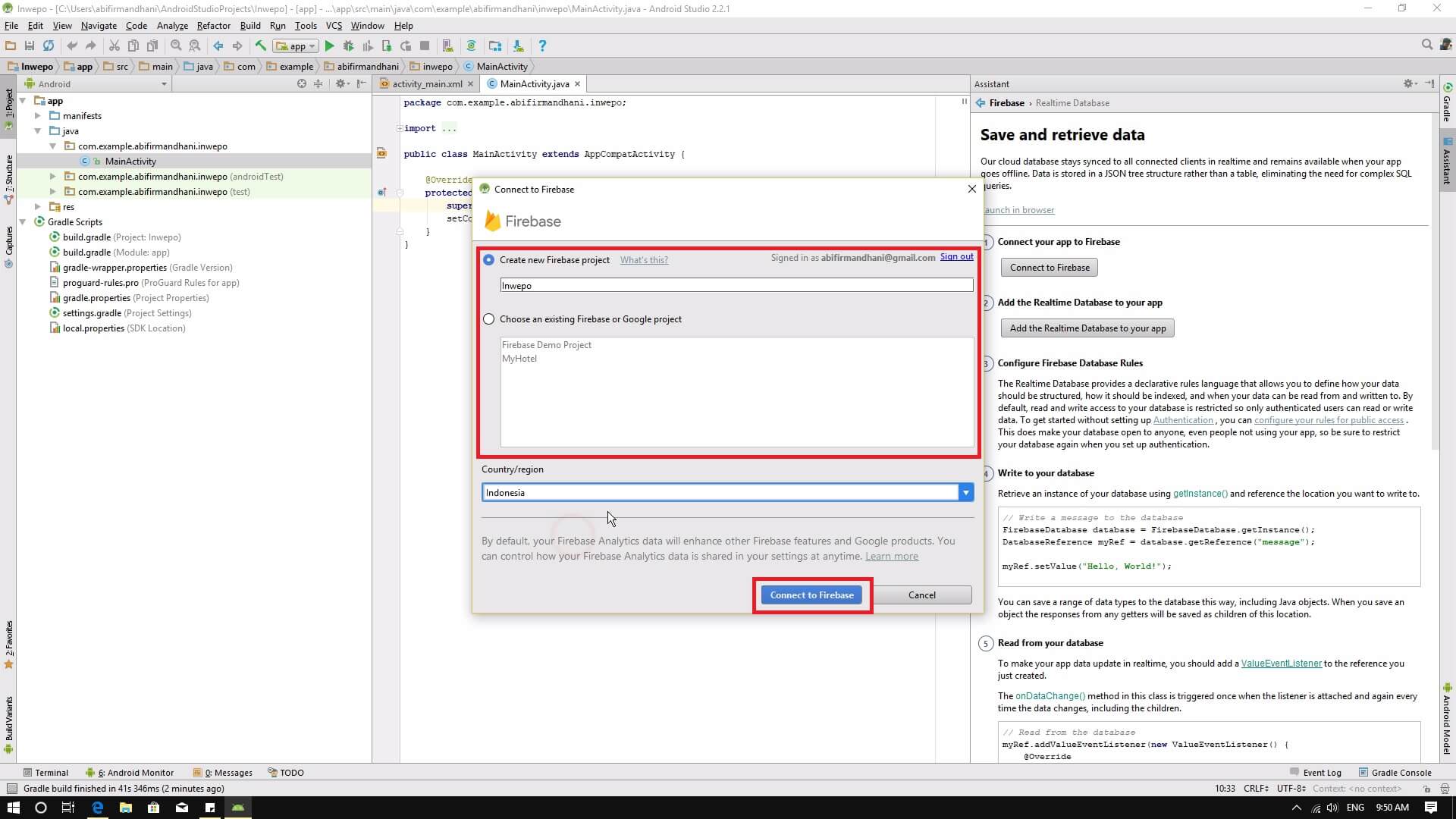
7. Akan muncul firebase *assistant*, dimana fungsinya adalah membantu dalam menggunakan firebase. Pilih ***realtime database***dan klik***save and retrieve data.***



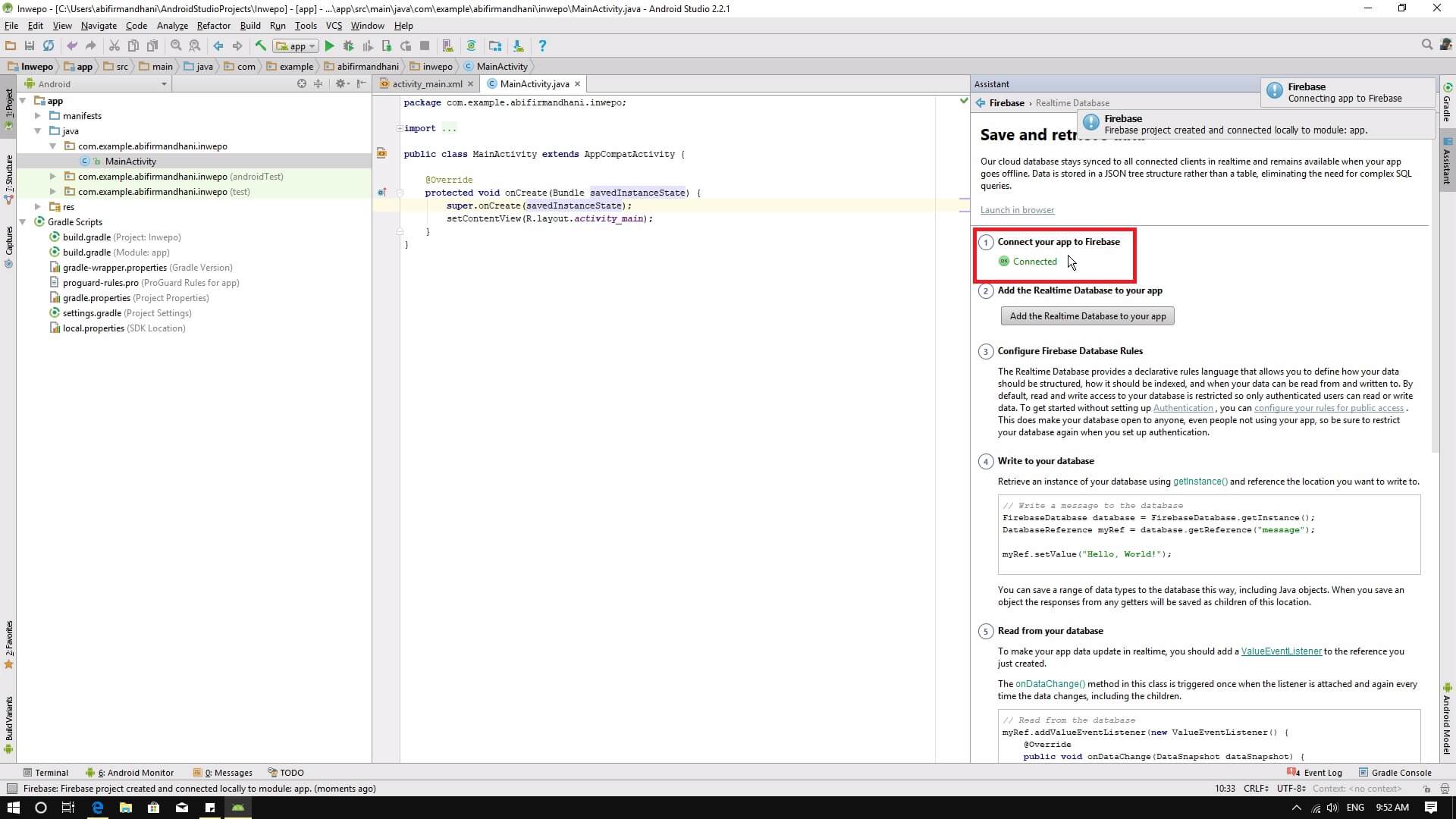
8. Firebase *assistant* akan memberikan beberapa langkah agar kamu bisa menggunakan *realtime database*. Langkah pertama adalah *connect* dengan firebase. Klik tombol***connect to* firebase**



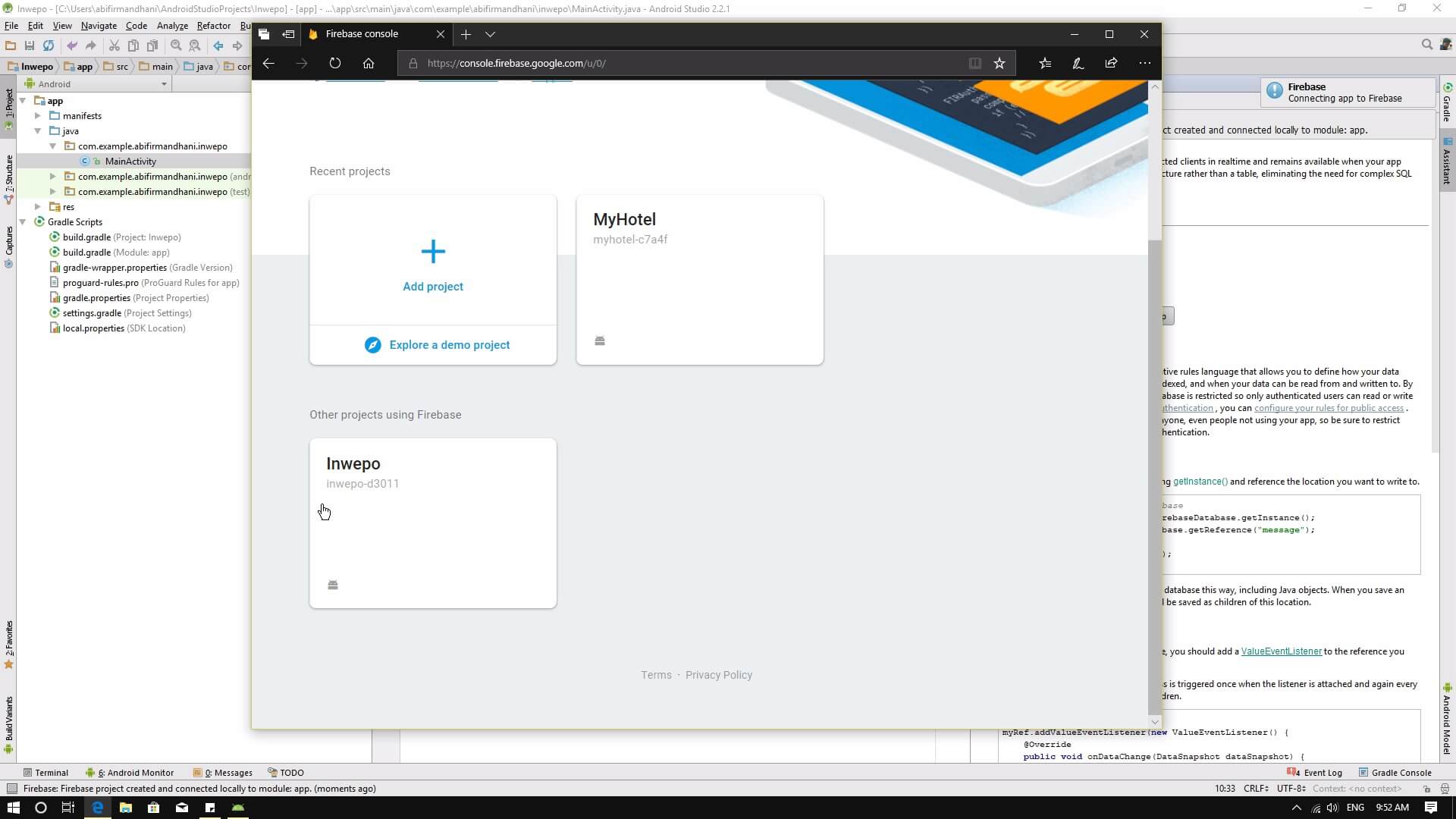
9. Lalu akan muncul jendela baru. Disini kamu bisa memilih antara membuat projek baru di firebase, atau menggunakan projek yang sudah ada. Jika kamu belum punya projek di firebase, pilih buat baru saja. Lalu Klik tombol***connect* to firebase.**



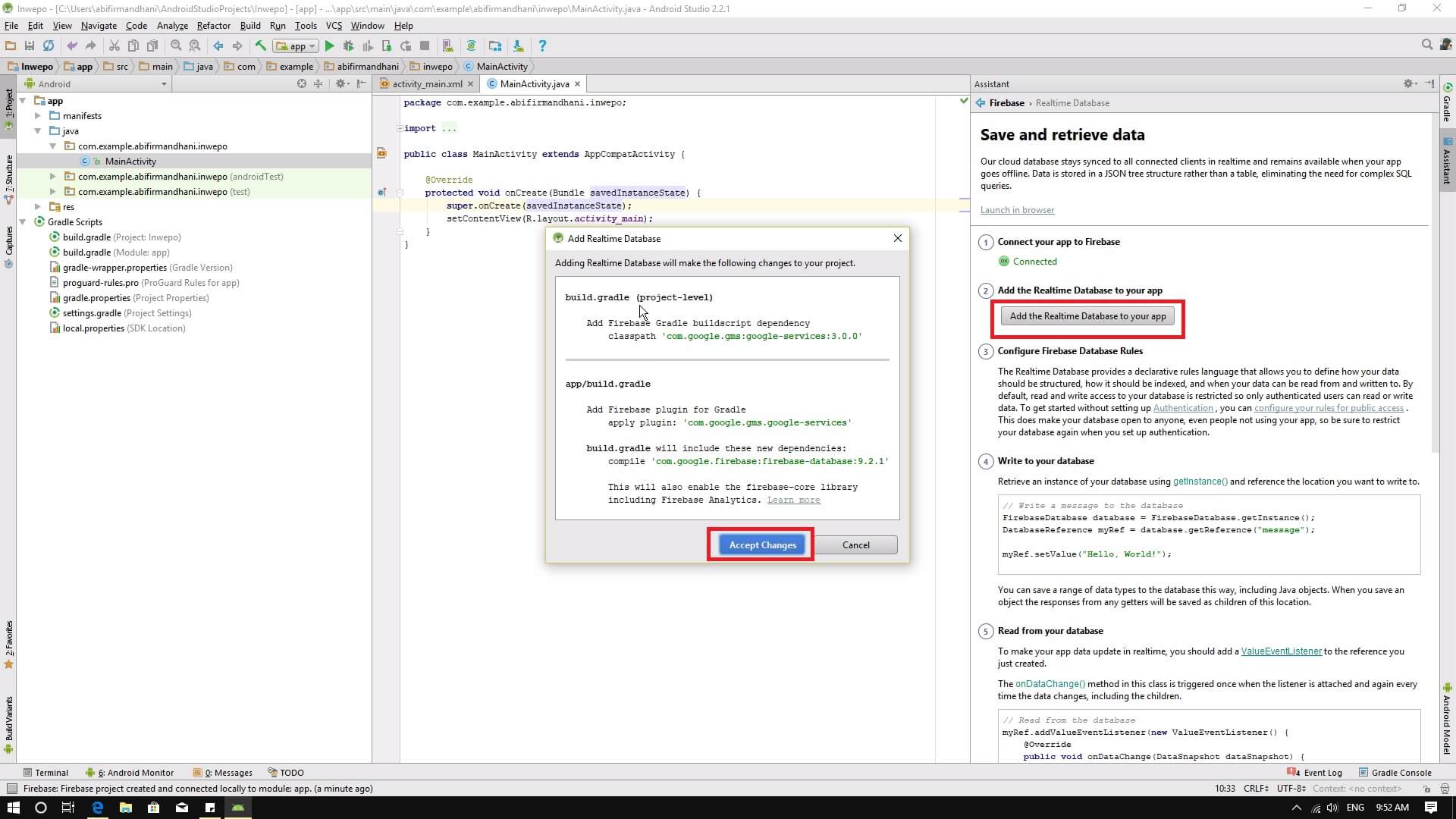
10. Berhasil terhubung dengan firebase.



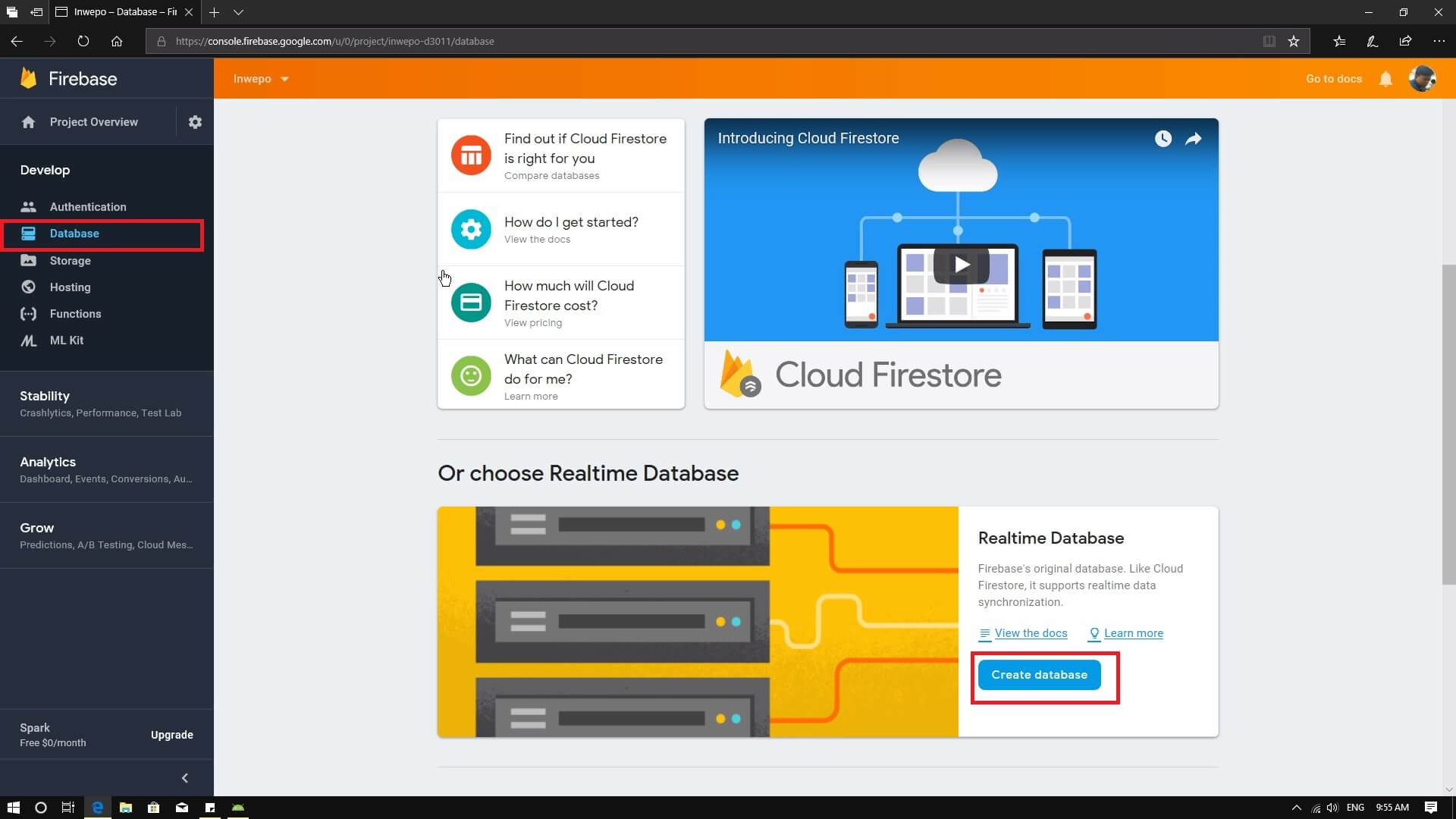
11. **Buka**[**Firebase console**](http://console.firebase.google.com/) dan pastikan akun google kamu sama dengan yang kamu pakai di Android studio. Bisa terlihat projek yang kamu buat pada langkah sebelumnya berhasil ditambah.



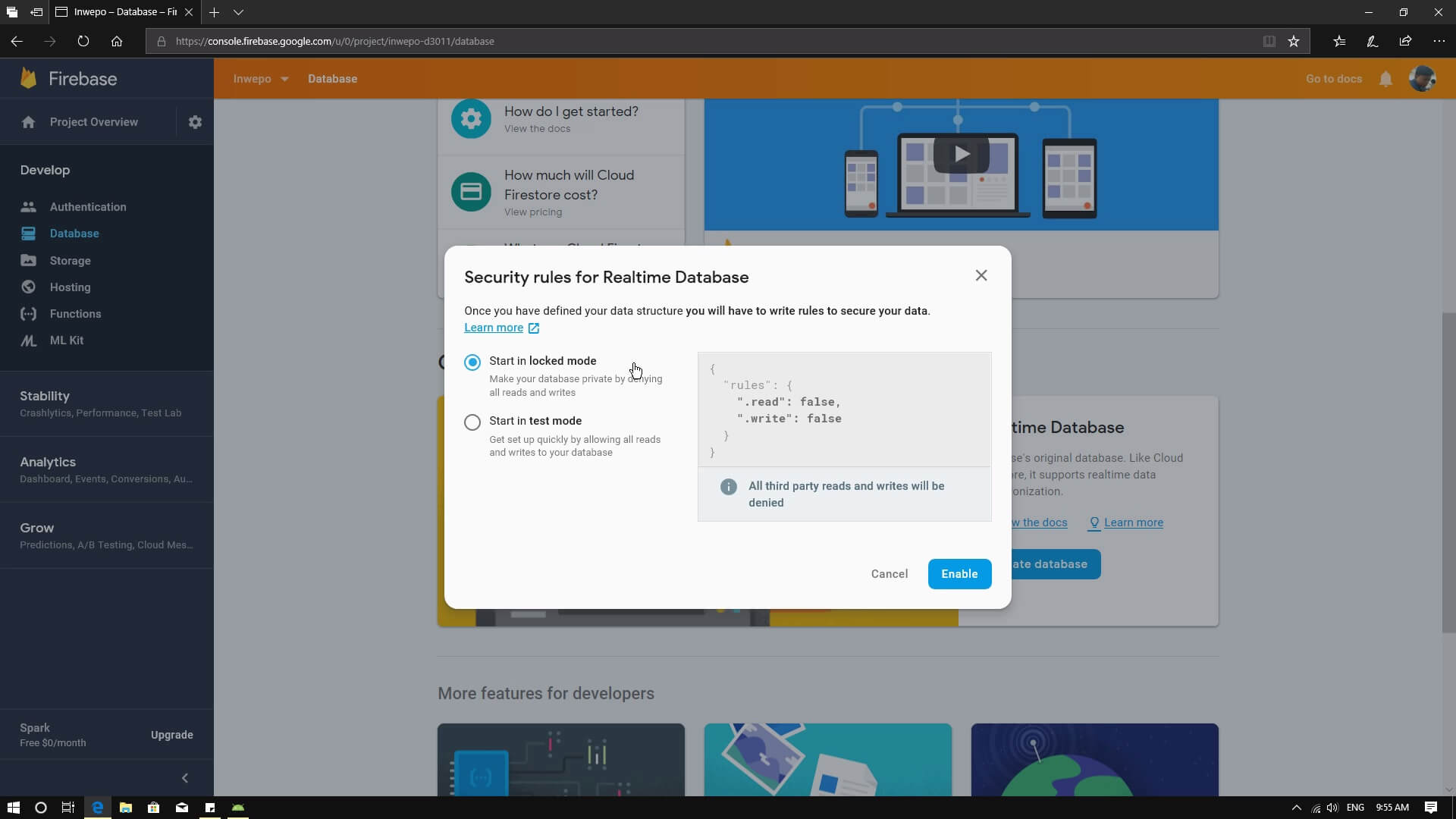
12. Lanjut ke langkah selanjutnya. Klik tombol***add realtime database to your app***. Pada langkah ini kamu akan menambah beberapa *tools* di *gradle* projek kamu, sehingga *realtime database* bisa berjalan di aplikasi kamu nantinya. Klik***accept changes*** untuk melanjutkan.



13. Buka projek kamu di firebase, dan buat database baru dengan cara klik tombol***create database***yang ada di menu database.



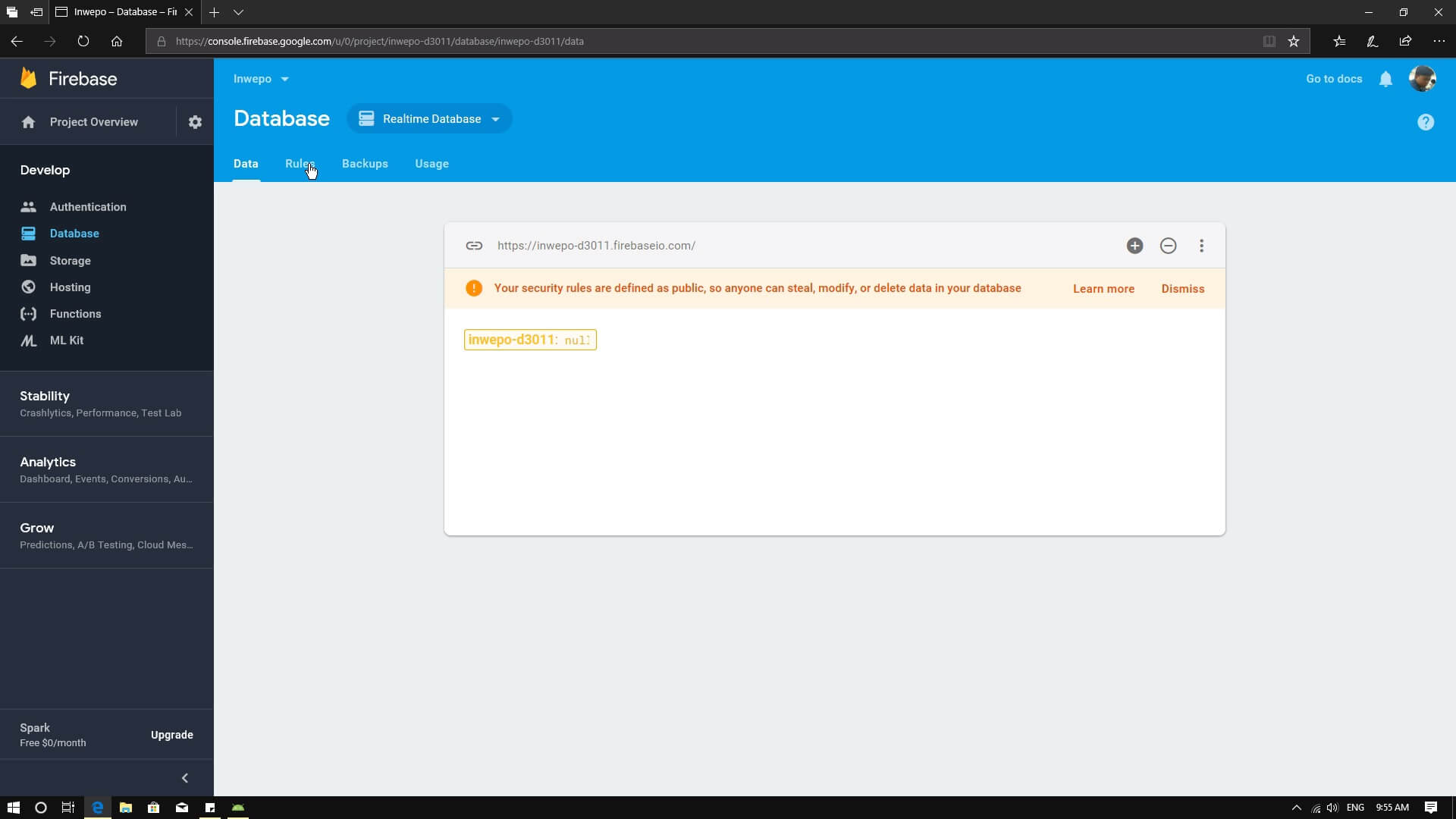
14. Saat kamu membuat *realtime database* kamu harus menentukan *rules* dari *database* tersebut.



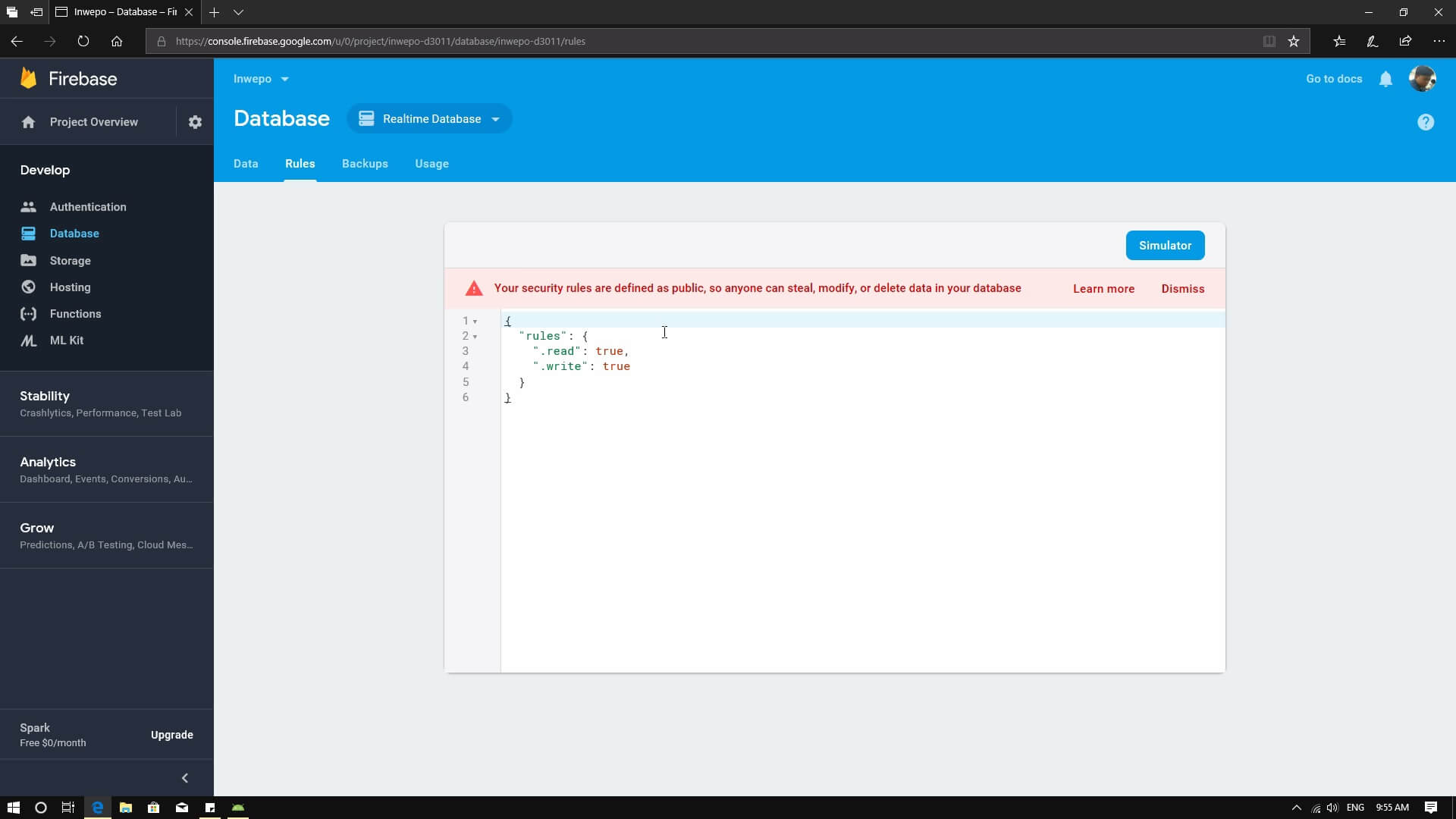
* Jika kamu memilih *locked mode*, maka *database* kamu tidak bisa di *read* dan *write* dari luar
* Jika kamu memilih *test mode*, maka *database* kamu bisa di *read*dan *write* dari luar

Pilih salah satu, dan klik tombol *enable*

15. Disini kamu berhasil membuat *database* baru dan datanya masih kosong.



16. Klik tab***Rules***untuk mengatur *rules* di *database* kamu.



Kamu bisa mengganti *rules database* kamu di sini.

Jika *true*, *maka read* dan *write* bisa dilakukan dari luar atau aplikasi lain. Sedangkan jika *false*, maka tidak bisa diakses dari luar

1. **Membuat Aplikasi POST CRUDS Integrasi Realtime Database menggunakan Firebase (Design)**
2. **Design Menampilkan Data**

Menampilkan semua data list dari Realtime Databases

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.swiperefreshlayout.widget.SwipeRefreshLayout  
 android:id="@+id/swipe\_refresh"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 <androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/rv"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />  
  
 </androidx.swiperefreshlayout.widget.SwipeRefreshLayout>  
  
 <LinearLayout  
 android:id="@+id/llcontainer"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:gravity="center"  
 android:orientation="vertical"  
 android:visibility="visible">  
  
 <ImageView  
 android:layout\_width="150dp"  
 android:layout\_height="150dp"  
 android:src="@drawable/empty\_data" />  
  
 <TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:text="@string/data\_empty"  
 android:textColor="#BAB4B4" />  
 </LinearLayout>  
  
 <com.google.android.material.floatingactionbutton.FloatingActionButton  
 android:id="@+id/fab"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentRight="true"  
 android:layout\_alignParentBottom="true"  
 android:layout\_marginRight="10dp"  
 android:layout\_marginBottom="10dp"  
 android:src="@drawable/ic\_baseline\_add\_24" />  
  
</RelativeLayout>

1. **Design Menambahkan Data**

<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <ImageView  
 android:id="@+id/iv\_empty\_image"  
 android:layout\_width="match\_parent"  
 android:layout\_height="200dp"  
 android:layout\_marginRight="20dp"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginTop="20dp"  
 android:src="@drawable/empty\_image" />  
  
 <EditText  
 android:id="@+id/et\_add\_note"  
 android:layout\_width="match\_parent"  
 android:padding="20dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginTop="30dp"  
 android:layout\_marginRight="20dp"  
 android:background="@drawable/ractangel\_edittext"  
 android:hint="Enter note here..."  
 android:paddingLeft="15dp"  
 android:paddingRight="15dp" />  
  
 <Button  
 android:id="@+id/btn\_save"  
 android:layout\_width="match\_parent"  
 android:layout\_height="60dp"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginTop="30dp"  
 android:layout\_marginRight="20dp"  
 android:background="@drawable/ractangel\_edittext\_disable"  
 android:enabled="false"  
 android:text="Save"  
 android:textAllCaps="false"  
 android:textColor="#B3A8A8"  
 android:textSize="18dp" />  
</LinearLayout>

1. **Design Item**

Menampilkan data peritem dari setiap tampilan yang ditampilkan

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:card\_view="http://schemas.android.com/apk/res-auto"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="vertical">  
  
 <androidx.cardview.widget.CardView  
 android:id="@+id/card\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_margin="10dp"  
 card\_view:cardCornerRadius="12dp"  
 card\_view:cardElevation="7dp">  
  
 <LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:background="?android:attr/selectableItemBackground"  
 android:orientation="vertical">  
  
 <ImageView  
 android:id="@+id/iv\_image"  
 android:layout\_width="match\_parent"  
 android:layout\_height="200dp"  
 android:scaleType="centerCrop"  
 android:src="@drawable/images" />  
  
 <TextView  
 android:id="@+id/desc\_text"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:padding="10dp"  
 android:text="Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. And the only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. As with all matters of the heart, you'll know when you find it." />  
 </LinearLayout>  
  
 </androidx.cardview.widget.CardView>  
</RelativeLayout>

1. **Membuat Aplikasi POST CRUDS Integrasi Realtime Database menggunakan Firebase (Java)**
2. **Class Constant**

Sebagai penampung dari parent realtime databases

public class Constants {  
 public static final String *POST* = "post";  
}

1. **Class Model Post**

Sebagai penampung sekaligus pengambilan data, dengan menerapkan setter dan getter

package com.tanwir.apppost;  
  
public class ModelPost {  
  
 private String id;  
 private String note;  
 private String image\_url;  
  
 public String getId() {  
 return id;  
 }  
  
 public void setId(String id) {  
 this.id = id;  
 }  
  
 public String getNote() {  
 return note;  
 }  
  
 public void setNote(String note) {  
 this.note = note;  
 }  
  
 public String getImage\_url() {  
 return image\_url;  
 }  
  
 public void setImage\_url(String image\_url) {  
 this.image\_url = image\_url;  
 }  
}

1. **Class Adapter Post**

Digunakan sebagai mengatur bagian menampilkan dataset ke dalam Recaycleview

package com.tanwir.apppost;  
  
import android.annotation.SuppressLint;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.ImageView;  
import android.widget.TextView;  
  
import androidx.annotation.NonNull;  
import androidx.recyclerview.widget.RecyclerView;  
  
import com.squareup.picasso.Picasso;  
  
import java.util.ArrayList;  
import java.util.List;  
  
  
public class AdapterPost extends RecyclerView.Adapter<AdapterPost.FavoritViewHolder> {  
  
 public static MClickListener *nListener*;  
 private List<ModelPost> list;  
  
 public AdapterPost(MClickListener listener) {  
 *nListener* = listener;  
 }  
  
 @NonNull  
 @SuppressLint("InflateParams")  
 @Override  
 public FavoritViewHolder onCreateViewHolder(ViewGroup p1, int p2) {  
 return new FavoritViewHolder(LayoutInflater.*from*(p1.getContext()).inflate(R.layout.*item\_post*, null));  
 }  
  
 @SuppressLint("SetTextI18n")  
 @Override  
 public void onBindViewHolder(final FavoritViewHolder holdr, int pos) {  
 ModelPost mp = list.get(pos);  
  
 Picasso.*get*().load(mp.getImage\_url()).into(holdr.iv\_image);  
 holdr.desc\_text.setText(mp.getNote());  
 }  
  
 @Override  
 public int getItemCount() {  
 return list != null ? list.size() : 0;  
 }  
  
 public void addModelPost(ModelPost op) {  
 list.add(op);  
 notifyDataSetChanged();  
 }  
  
 public ModelPost getModelPost(int position) {  
 return list.get(position);  
 }  
  
 public static class FavoritViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener {  
  
 ImageView iv\_image;  
 TextView desc\_text;  
  
 public FavoritViewHolder(@NonNull View view) {  
 super(view);  
 iv\_image = view.findViewById(R.id.*iv\_image*);  
 desc\_text = view.findViewById(R.id.*desc\_text*);  
 itemView.setOnClickListener(this);  
 }  
  
 @Override  
 public void onClick(View view) {  
 *nListener*.onClick(getLayoutPosition());  
 }  
 }  
  
 public interface MClickListener {  
 void onClick(int position);  
 }  
}

1. **Class Add Post**

Berfungsi sebagai menambahkan data ke Realtime Databases

package com.tanwir.apppost;  
  
import android.annotation.SuppressLint;  
import android.app.ProgressDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.net.Uri;  
import android.os.Bundle;  
import android.text.Editable;  
import android.text.TextWatcher;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.app.AppCompatDelegate;  
  
import com.google.android.gms.tasks.OnSuccessListener;  
import com.google.android.gms.tasks.Task;  
import com.google.firebase.FirebaseApp;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.annotations.NotNull;  
import com.google.firebase.storage.FirebaseStorage;  
import com.google.firebase.storage.StorageReference;  
import com.google.firebase.storage.UploadTask;  
import com.theartofdev.edmodo.cropper.CropImage;  
import com.theartofdev.edmodo.cropper.CropImageView;  
  
import java.io.FileNotFoundException;  
import java.io.InputStream;  
import java.util.Objects;  
  
public class AddPost extends AppCompatActivity {  
  
 private ImageView iv\_image;  
 private EditText et\_note;  
 private Button btn\_save;  
  
 private DatabaseReference mDReference;  
 private FirebaseDatabase mFInstance;  
 private StorageReference sReference;  
  
 private Uri resultUriImage;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.add\_post);  
 AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE\_NIGHT\_NO);  
 setTitle("ADD POST");  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setDisplayShowHomeEnabled(true);  
  
 init();  
 initFirebase();  
 }  
  
 private void init() {  
  
 iv\_image = findViewById(R.id.iv\_image);  
 et\_note = findViewById(R.id.et\_add\_note);  
 btn\_save = findViewById(R.id.btn\_save);  
  
 iv\_image.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 CropImage.activity()  
 .setGuidelines(CropImageView.Guidelines.ON)  
 .setRequestedSize(500, 250)  
 .start(AddPost.this);  
 }  
 });  
  
 et\_note.addTextChangedListener(new TextWatcher() {  
 @Override  
 public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {  
  
 }  
  
 @Override  
 public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {  
 if (charSequence.length() != 0) {  
 btn\_save.setEnabled(true);  
 btn\_save.setBackgroundResource(R.drawable.ractangel\_edittext\_color);  
 btn\_save.setTextColor(getResources().getColor(R.color.*white*));  
 } else {  
 btn\_save.setEnabled(false);  
 btn\_save.setBackgroundResource(R.drawable.*ractangel\_edittext\_disable*);  
 btn\_save.setTextColor(getResources().getColor(R.color.*gray*));  
 }  
 }  
  
 @SuppressLint("SetTextI18n")  
 @Override  
 public void afterTextChanged(Editable editable) {  
 }  
 });  
  
 btn\_save.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 dialogSavePost();  
 }  
 });  
 }  
  
 private void initFirebase() {  
 FirebaseApp.initializeApp(this);  
 mFInstance = FirebaseDatabase.getInstance();  
 mDReference = mFInstance.getReference(Constants.POST);  
 sReference = FirebaseStorage.getInstance().getReference(Constants.POST);  
 }  
  
 private void dialogSavePost() {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Sure want to save!");  
  
 builder.setPositiveButton("Save", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int id) {  
 savePost();  
 }  
 });  
  
 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.dismiss();  
 }  
 });  
 builder.show();  
 }  
  
 private void savePost() {  
 if (resultUriImage != null) {  
 final ProgressDialog progressDialog = new ProgressDialog(this);  
 progressDialog.show();  
  
 ModelPost post = new ModelPost();  
 post.setId(mDReference.push().getKey());  
  
 UploadTask uploadTask = sReference.child(post.getId()).putFile(resultUriImage);  
 uploadTask.addOnProgressListener(taskSnapshot -> {  
  
 double progress = (100.0 \* taskSnapshot.getBytesTransferred()) / taskSnapshot.getTotalByteCount();  
 progressDialog.setMessage("Please wait " + ((int) progress) + "%...");  
  
 }).addOnPausedListener(taskSnapshot -> {  
 progressDialog.dismiss();  
 System.out.println("Upload is paused");  
  
 }).addOnFailureListener(exception -> {  
 progressDialog.dismiss();  
 Toast.makeText(this, "Process failed", Toast.LENGTH\_SHORT).show();  
 }).addOnSuccessListener(taskSnapshot -> {  
 progressDialog.dismiss();  
  
 Task<Uri> uri = taskSnapshot.getStorage().getDownloadUrl();  
 while (!uri.isComplete()) ;  
 Uri url = uri.getResult();  
  
 post.setNote(et\_note.getText().toString());  
 post.setImage\_url(url.toString());  
  
// save db  
 mDReference.child(post.getId())  
 .setValue(post).addOnSuccessListener(this, new OnSuccessListener<Void>() {  
 @Override  
 public void onSuccess(Void mVoid) {  
 progressDialog.dismiss();  
 Toast.makeText(AddPost.this, "saved successfully!", Toast.LENGTH\_LONG).show();  
 finish();  
 }  
 });  
 });  
  
 } else {  
 Toast.makeText(this, "Image not entered", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @SuppressLint("MissingSuperCall")  
 @Override  
 public void onActivityResult(int requestCode, int resultCode, Intent data) {  
 if (requestCode == CropImage.CROP\_IMAGE\_ACTIVITY\_REQUEST\_CODE) {  
 CropImage.ActivityResult result = CropImage.getActivityResult(data);  
 if (resultCode == RESULT\_OK) {  
 resultUriImage = result.getUri();  
 try {  
 InputStream inputStream = getContentResolver().openInputStream(resultUriImage);  
 Bitmap bitmap = BitmapFactory.decodeStream(inputStream);  
 iv\_image.setImageBitmap(bitmap);  
  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 }  
 }  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 if (item.getItemId() == android.R.id.home) {  
 finish();  
 }  
 return super.onOptionsItemSelected(item);  
 }  
}

1. **Class Update Post**

Berfungsi sebagai ubah data

package com.tanwir.apppost;  
  
import android.annotation.SuppressLint;  
import android.app.ProgressDialog;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.net.Uri;  
import android.os.Bundle;  
import android.text.Editable;  
import android.text.TextWatcher;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.ImageView;  
import android.widget.Toast;  
  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.app.AppCompatDelegate;  
  
import com.google.android.gms.tasks.OnSuccessListener;  
import com.google.android.gms.tasks.Task;  
import com.google.firebase.FirebaseApp;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.storage.FirebaseStorage;  
import com.google.firebase.storage.StorageReference;  
import com.google.firebase.storage.UploadTask;  
import com.squareup.picasso.Picasso;  
import com.theartofdev.edmodo.cropper.CropImage;  
import com.theartofdev.edmodo.cropper.CropImageView;  
  
import java.io.FileNotFoundException;  
import java.io.InputStream;  
  
public class UpdatePost extends AppCompatActivity {  
  
 private ImageView iv\_image;  
 private EditText et\_note;  
 private Button btn\_save;  
  
 private DatabaseReference mDReference;  
 private FirebaseDatabase mFInstance;  
 private StorageReference sReference;  
  
 private Uri resultUriImage;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.add\_post);  
 AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE\_NIGHT\_NO);  
 setTitle("UPDATE POST");  
 getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
 getSupportActionBar().setDisplayShowHomeEnabled(true);  
  
 init();  
 initFirebase();  
 }  
  
 @SuppressLint("SetTextI18n")  
 private void init() {  
  
 iv\_image = findViewById(R.id.iv\_image);  
 et\_note = findViewById(R.id.et\_add\_note);  
 btn\_save = findViewById(R.id.btn\_save);  
 btn\_save.setText("Update");  
  
 Picasso.get().load(getIntent().getStringExtra("image\_url")).into(iv\_image);  
 et\_note.setText(getIntent().getStringExtra("note"));  
  
 iv\_image.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 CropImage.activity()  
 .setGuidelines(CropImageView.Guidelines.ON)  
 .setRequestedSize(500, 250)  
 .start(UpdatePost.this);  
 }  
 });  
  
 et\_note.addTextChangedListener(new TextWatcher() {  
 @Override  
 public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {  
  
 }  
  
 @Override  
 public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {  
 if (charSequence.length() != 0) {  
 btn\_save.setEnabled(true);  
 btn\_save.setBackgroundResource(R.drawable.ractangel\_edittext\_color);  
 btn\_save.setTextColor(getResources().getColor(R.color.white));  
 } else {  
 btn\_save.setEnabled(false);  
 btn\_save.setBackgroundResource(R.drawable.ractangel\_edittext\_disable);  
 btn\_save.setTextColor(getResources().getColor(R.color.gray));  
 }  
 }  
  
 @SuppressLint("SetTextI18n")  
 @Override  
 public void afterTextChanged(Editable editable) {  
 }  
 });  
  
 btn\_save.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 dialogUpdatePost();  
 }  
 });  
 }  
  
 private void initFirebase() {  
 FirebaseApp.initializeApp(this);  
 mFInstance = FirebaseDatabase.getInstance();  
 mDReference = mFInstance.getReference(Constants.POST).child(getIntent().getStringExtra("id"));  
 sReference = FirebaseStorage.getInstance().getReference(Constants.POST).child(getIntent().getStringExtra("id"));  
 }  
  
 private void dialogUpdatePost() {  
 AlertDialog.Builder builder = new AlertDialog.Builder(this);  
 builder.setTitle("Sure want to update!");  
  
 builder.setPositiveButton("Update", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int id) {  
 updatePost();  
 }  
 });  
  
 builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int id) {  
 dialog.dismiss();  
 }  
 });  
 builder.show();  
 }  
  
 private void updatePost() {  
 if (resultUriImage != null) {  
 final ProgressDialog progressDialog = new ProgressDialog(this);  
 progressDialog.show();  
  
 ModelPost post = new ModelPost();  
 post.setId(getIntent().getStringExtra("id"));  
  
 UploadTask uploadTask = sReference.putFile(resultUriImage);  
 uploadTask.addOnProgressListener(taskSnapshot -> {  
  
 double progress = (100.0 \* taskSnapshot.getBytesTransferred()) / taskSnapshot.getTotalByteCount();  
 progressDialog.setMessage("Please wait " + ((int) progress) + "%...");  
  
 }).addOnPausedListener(taskSnapshot -> {  
 progressDialog.dismiss();  
 System.out.println("Upload is paused");  
  
 }).addOnFailureListener(exception -> {  
 progressDialog.dismiss();  
 Toast.makeText(this, "Process failed", Toast.LENGTH\_SHORT).show();  
 }).addOnSuccessListener(taskSnapshot -> {  
 progressDialog.dismiss();  
  
 Task<Uri> uri = taskSnapshot.getStorage().getDownloadUrl();  
 while (!uri.isComplete()) ;  
 Uri url = uri.getResult();  
  
 post.setNote(et\_note.getText().toString());  
 post.setImage\_url(url.toString());  
  
// update db  
 mDReference.setValue(post).addOnSuccessListener(this, new OnSuccessListener<Void>() {  
 @Override  
 public void onSuccess(Void mVoid) {  
 progressDialog.dismiss();  
 Toast.makeText(UpdatePost.this, "Update successfully!", Toast.LENGTH\_LONG).show();  
 finish();  
 }  
 });  
 });  
  
 } else {  
 Toast.makeText(this, "Image not entered", Toast.LENGTH\_SHORT).show();  
 }  
 }  
  
 @SuppressLint("MissingSuperCall")  
 @Override  
 public void onActivityResult(int requestCode, int resultCode, Intent data) {  
 if (requestCode == CropImage.CROP\_IMAGE\_ACTIVITY\_REQUEST\_CODE) {  
 CropImage.ActivityResult result = CropImage.getActivityResult(data);  
 if (resultCode == RESULT\_OK) {  
 resultUriImage = result.getUri();  
 try {  
 InputStream inputStream = getContentResolver().openInputStream(resultUriImage);  
 Bitmap bitmap = BitmapFactory.decodeStream(inputStream);  
 iv\_image.setImageBitmap(bitmap);  
  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 }  
  
 }  
 }  
 }  
  
 @Override  
 public boolean onOptionsItemSelected(MenuItem item) {  
 if (item.getItemId() == android.R.id.home) {  
 finish();  
 }  
 return super.onOptionsItemSelected(item);  
 }  
}

1. **Class MainActivity**

Berfungsi sebagai menampilkan data

package com.tanwir.apppost;  
  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.appcompat.app.AppCompatDelegate;  
import androidx.appcompat.widget.SearchView;  
import androidx.recyclerview.widget.LinearLayoutManager;  
import androidx.recyclerview.widget.RecyclerView;  
import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;  
  
import android.app.ProgressDialog;  
import android.app.SearchManager;  
import android.content.Context;  
import android.content.Intent;  
import android.os.Bundle;  
import android.view.Menu;  
import android.view.MenuInflater;  
import android.view.MenuItem;  
import android.view.View;  
import android.widget.LinearLayout;  
import android.widget.Toast;  
  
import com.google.android.gms.tasks.OnSuccessListener;  
import com.google.firebase.FirebaseApp;  
import com.google.firebase.database.DataSnapshot;  
import com.google.firebase.database.DatabaseError;  
import com.google.firebase.database.DatabaseReference;  
import com.google.firebase.database.FirebaseDatabase;  
import com.google.firebase.database.Query;  
import com.google.firebase.database.ValueEventListener;  
import com.google.firebase.storage.FirebaseStorage;  
import com.google.firebase.storage.StorageReference;  
  
import java.util.ArrayList;  
  
public class MainActivity extends AppCompatActivity implements  
 SwipeRefreshLayout.OnRefreshListener,  
 AdapterPost.MClickListener {  
  
 private RecyclerView rv;  
 private SwipeRefreshLayout swipe;  
 private ProgressDialog prgDialog;  
 private AdapterPost adapterPost;  
  
 private DatabaseReference mDReference;  
 private FirebaseDatabase mFInstance;  
 private StorageReference sReference;  
  
 private LinearLayout llcontainer;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 AppCompatDelegate.*setDefaultNightMode*(AppCompatDelegate.*MODE\_NIGHT\_NO*);  
 setTitle("POST LIST");  
  
 init();  
 initFirebase();  
 }  
  
 private void init() {  
 llcontainer = findViewById(R.id.*llcontainer*);  
  
 rv = findViewById(R.id.*rv*);  
 swipe = findViewById(R.id.*swipe\_refresh*);  
 swipe.setColorSchemeColors(getResources().getColor(android.R.color.*white*), getResources().getColor(android.R.color.*white*), getResources().getColor(android.R.color.*white*));  
 swipe.setProgressBackgroundColorSchemeColor(getResources().getColor(R.color.*purple\_200*));  
 swipe.setOnRefreshListener(this);  
 swipe.post(() -> {  
 swipe.setRefreshing(true);  
 readData();  
 }  
 );  
  
 findViewById(R.id.*fab*).setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 startActivity(new Intent(MainActivity.this, AddPost.class));  
 }  
 });  
  
 }  
  
 private void initFirebase() {  
 FirebaseApp.*initializeApp*(this);  
 mFInstance = FirebaseDatabase.*getInstance*();  
 mDReference = mFInstance.getReference(Constants.*POST*);  
 sReference = FirebaseStorage.*getInstance*().getReference(Constants.*POST*);  
 }  
  
 @Override  
 public void onRefresh() {  
 readData();  
 }  
  
 private void readData() {  
 configRecycle();  
  
 mDReference.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(DataSnapshot dataSnapshot) {  
  
 if (dataSnapshot.getChildrenCount() > 0) {  
 for (DataSnapshot mDataSnapshot : dataSnapshot.getChildren()) {  
 ModelPost mp = mDataSnapshot.getValue(ModelPost.class);  
 adapterPost.addModelPost(mp);  
 }  
  
 llcontainer.setVisibility(View.*GONE*);  
 } else {  
 llcontainer.setVisibility(View.*VISIBLE*);  
 }  
  
 swipe.setRefreshing(false);  
 }  
  
 @Override  
 public void onCancelled(DatabaseError databaseError) {  
 Toast.*makeText*(MainActivity.this, databaseError.getDetails() + " " + databaseError.getMessage(), Toast.*LENGTH\_LONG*).show();  
 swipe.setRefreshing(false);  
 }  
 });  
 }  
  
 public void configRecycle() {  
 rv.setHasFixedSize(true);  
 rv.setLayoutManager(new LinearLayoutManager(this));  
 adapterPost = new AdapterPost(this);  
 rv.setAdapter(adapterPost);  
 }  
  
 @Override  
 protected void onRestart() {  
 super.onRestart();  
 readData();  
 }  
  
 @Override  
 public void onClick(int position) {  
 ModelPost modelPost = adapterPost.getModelPost(position);  
  
 final AlertDialog.Builder builder = new AlertDialog.Builder(this, R.style.*AlertDialogStyle*);  
 builder.setTitle("Choose Action");  
 builder.setItems(new String[]{"Update", "Delete"}, (dialog, item) -> {  
 if (item == 0) {  
// *TODO: UPDATE* Intent in = new Intent(MainActivity.this, UpdatePost.class);  
 in.putExtra("id", modelPost.getId());  
 in.putExtra("image\_url", modelPost.getImage\_url());  
 in.putExtra("note", modelPost.getNote());  
 startActivity(in);  
  
 } else {  
// *TODO: DELETE* final android.app.AlertDialog.Builder builder2 = new android.app.AlertDialog.Builder(this);  
 builder2.setMessage("Sure you want to delete");  
 builder2.setPositiveButton("Yes", (dialogInterface, ii) -> {  
  
 if (mDReference != null) {  
 prgDialog = new ProgressDialog(this, R.style.*MyAlertDialogStyle*);  
 prgDialog.setMessage("Process delete...");  
 prgDialog.setCancelable(false);  
 prgDialog.show();  
  
 mDReference.child(modelPost.getId()).removeValue().addOnSuccessListener(new OnSuccessListener<Void>() {  
 @Override  
 public void onSuccess(Void mVoid) {  
 // delete file storage  
 sReference.child(modelPost.getId()).delete();  
  
 Toast.*makeText*(MainActivity.this, "Successfully deleted", Toast.*LENGTH\_LONG*).show();  
 prgDialog.dismiss();  
 readData();  
 }  
 });  
 }  
  
 });  
 builder2.setNegativeButton("No", (dialogInterface, i) -> dialogInterface.dismiss());  
 builder2.show();  
  
 }  
 });  
 builder.show();  
 }  
  
 private void searchPost(String qry) {  
 configRecycle();  
  
 Query query = mDReference.orderByChild("note").startAt(qry).endAt(qry+"\uf8ff");  
 query.addValueEventListener(new ValueEventListener() {  
 @Override  
 public void onDataChange(DataSnapshot dataSnapshot) {  
 for (DataSnapshot mDataSnapshot : dataSnapshot.getChildren()) {  
 ModelPost mp = mDataSnapshot.getValue(ModelPost.class);  
 adapterPost.addModelPost(mp);  
 }  
 swipe.setRefreshing(false);  
 }  
  
 @Override  
 public void onCancelled(DatabaseError databaseError) {  
 Toast.*makeText*(MainActivity.this, databaseError.getDetails() + " " + databaseError.getMessage(), Toast.*LENGTH\_LONG*).show();  
 swipe.setRefreshing(false);  
 }  
 });  
 }  
  
 @Override  
 public boolean onCreateOptionsMenu(Menu menu) {  
 getMenuInflater().inflate(R.menu.*search\_post*, menu);  
  
 MenuItem myActionMenuItem = menu.findItem(R.id.*action\_search*);  
 SearchView searchView = (SearchView) myActionMenuItem.getActionView();  
 searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
 @Override  
 public boolean onQueryTextSubmit(String query) {  
 searchPost(query);  
 if (!searchView.isIconified()) {  
 searchView.setIconified(true);  
 }  
 myActionMenuItem.collapseActionView();  
 return false;  
  
 }  
  
 @Override  
 public boolean onQueryTextChange(String s) {  
 // UserFeedback.show( "SearchOnQueryTextChanged: " + s);  
 return false;  
 }  
 });  
 return true;  
 }  
}