



Firestore Realtime Database

In this Lecture, you will learn:

- Firebase Introduction
- Creating Firebase Project
- Linking project with Firebase

Firestore is a backend service to develop high-quality apps. Here we will explore the nosql database feature of Firebase. Firebase offers services including storage, realtime database, authentication etc. It also gives an analysis of usage of these services along with the details of users using it.

You can integrate Firebase services in your Android app directly from Android Studio using the Assistant window. Make sure you have installed Google Repository version 26 or higher, with following steps:

Click Tools > SDK Manager.

Click the SDK Tools tab.

Check the Google Repository checkbox, and click OK.

Click OK to install.

Use the Assistant window in Android Studio by following steps:


Click Tools > Firebase to open the Assistant window.

Click to expand one of the listed features (for example, Realtime Database), then click the Get Started tutorial to connect to Firebase and add the necessary code to your app.






Steps to add Firebase to an Android app using Firebase Assistant:

1. Open the android studio and click on **Tools** in the upper left corner.
2. Click on the **Firestore** option in the drop down menu.
3. A menu will appear on the right side of screen. It will show services that Firestore offers. Choose the desired service.
4. Click **Connect to Firestore** option in the menu of desired service.
5. Add the dependencies of your service by clicking on the Add [YOUR SERVICE NAME] to the app option.


Assistant

 **Firebase**

Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. [Learn more](#)

-  **Analytics**
Measure user activity and engagement with free, easy, and unlimited analytics. [More info](#)
-  **Cloud Messaging**
Deliver and receive messages and notifications reliably across cloud and device. [More info](#)
-  **Authentication**
Sign in and manage users with ease, accepting emails, Google Sign-In, Facebook and other login providers. [More info](#)
-  **Realtime Database**
Store and sync data in realtime across all connected clients. [More info](#)
-  **Storage**
Store and retrieve large files like images, audio, and video without writing server-side code. [More info](#)

Assistant

 **Firebase** > Realtime Database

Save and retrieve data

Our cloud database stays synced to all connected clients in realtime and remains available when your app goes offline. Data is stored in a JSON tree structure rather than a table, eliminating the need for complex SQL queries.

[Launch in browser](#)

- 1 Connect your app to Firebase**
[Connect to Firebase](#)
- 2 Add the Realtime Database to your app**
[Add the Realtime Database to your app](#)
- 3 Configure Firebase Database Rules**
The Realtime Database provides a declarative rules language that allows you to define how your data should be structured, how it should be

After adding Firebase Realtime Database dependency, synchronize the project. This may result in errors in the app/build.gradle file. To resolve the errors make sure that a valid version of the Database is added (example: implementation 'com.google.firebase:firebase-database:16.0.1'). With this, you will also need another dependency:

```
implementation 'com.google.firebase:firebase-core:16.0.3'
```

The app/build.gradle should look like:

```
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'
android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "com.example.shan.fbbaseapp1"
        minSdkVersion 23
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
}
```

```

        buildTypes {
            release {
                minifyEnabled false
                proguardFiles getDefaultProguardFile('proguard-
android.txt'), 'proguard-rules.pro'
            }
        }
    }
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support.appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-
layout:1.1.3'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation
'com.android.support.test.espresso:espresso-core:3.0.2'
    implementation 'com.google.firebase:firebase-database:16.0.1'
    implementation 'com.google.firebase:firebase-core:16.0.3'
}

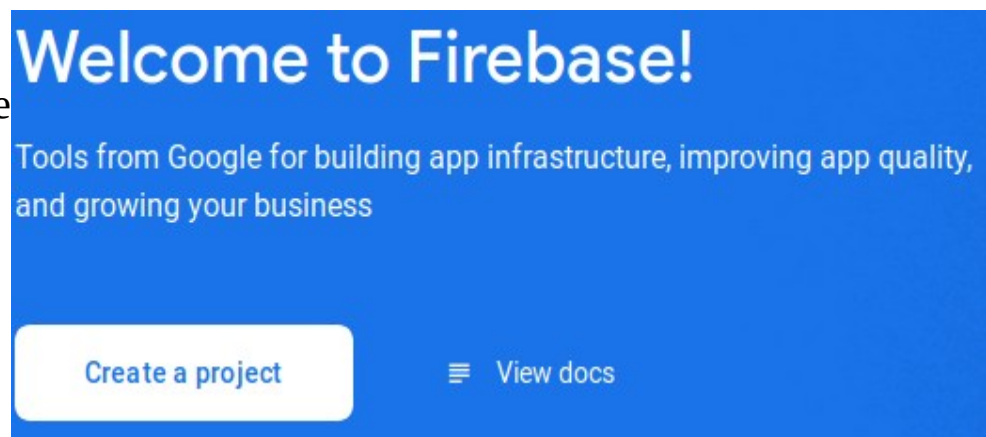
```

Before finally synchronizing the project, you will need to add the google-services.json configuration file downloaded from Firebase console. The steps to create and download this file are mentioned next.

Creating Firebase Project

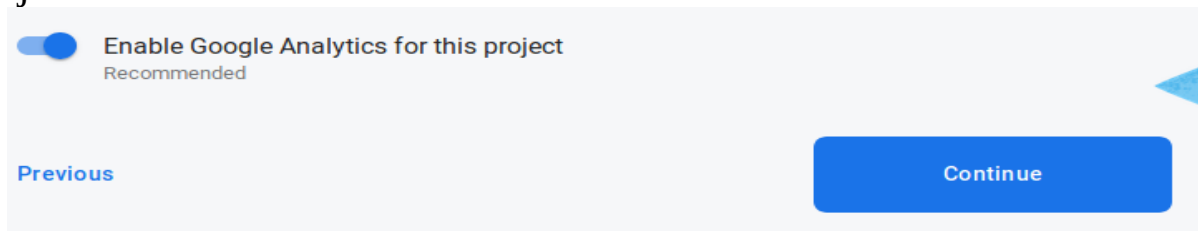
Open Firebase Console at <https://console.firebase.google.com> and create a firebase account to start with. In Firebase console Create a New Project by clicking on the “**Create New Project**” Button as shown below.

Now you will see
Step 1 of 3.

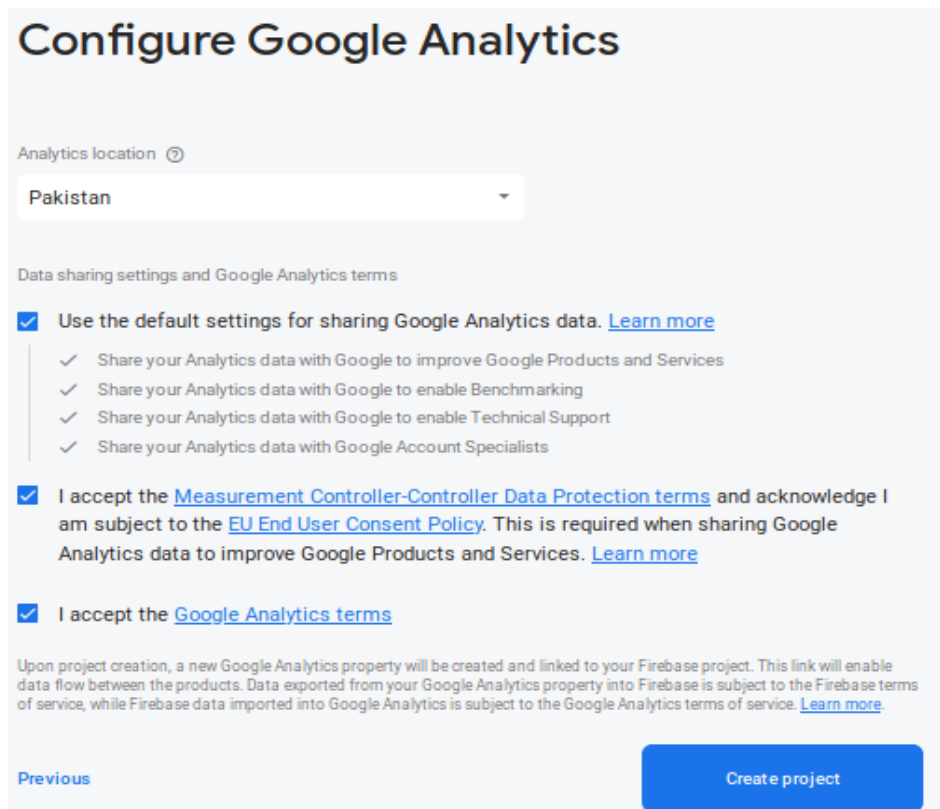


1. In first step, name the project and accept the Firebase terms and click “Continue” button.

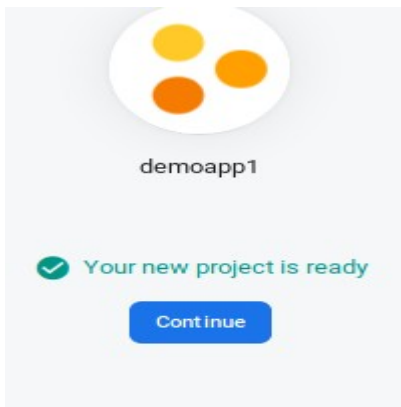
2. Second step is Google Analytics for your Firebase project. You can either leave it default enabled or disable it and click Continue. If you disable the Analytics, the third step will not appear and the button will be Create project. In our case, we will just disable the Analytics and create project.



3. If you enabled Analytics, the third step asks your Country and terms and conditions of Analytics. Accept these terms and click Create project.

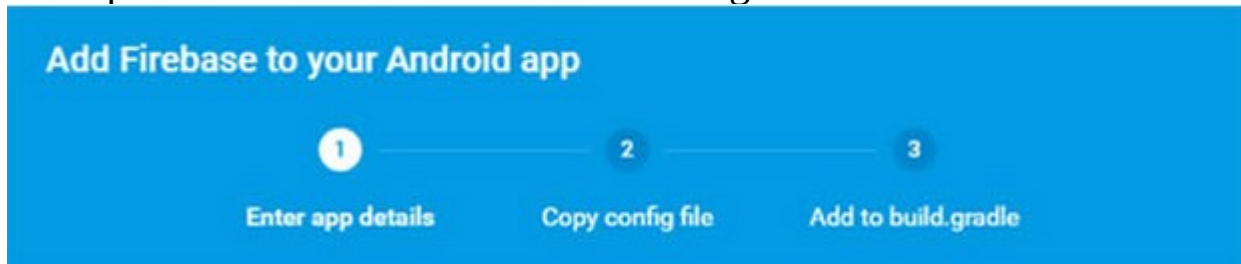


Next screen shows your project is ready. Click Continue.

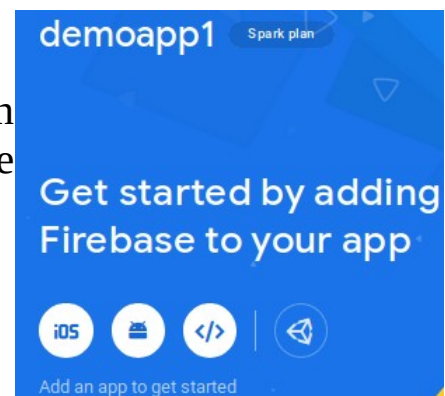


Linking Android Application with the Firebase Project

The steps involved are summarized in the figure below:



On the Project main page, you will be given options to Add an app for the project. Choose the Android icon as shown in the figure.



You would need to register your app with Firebase project.

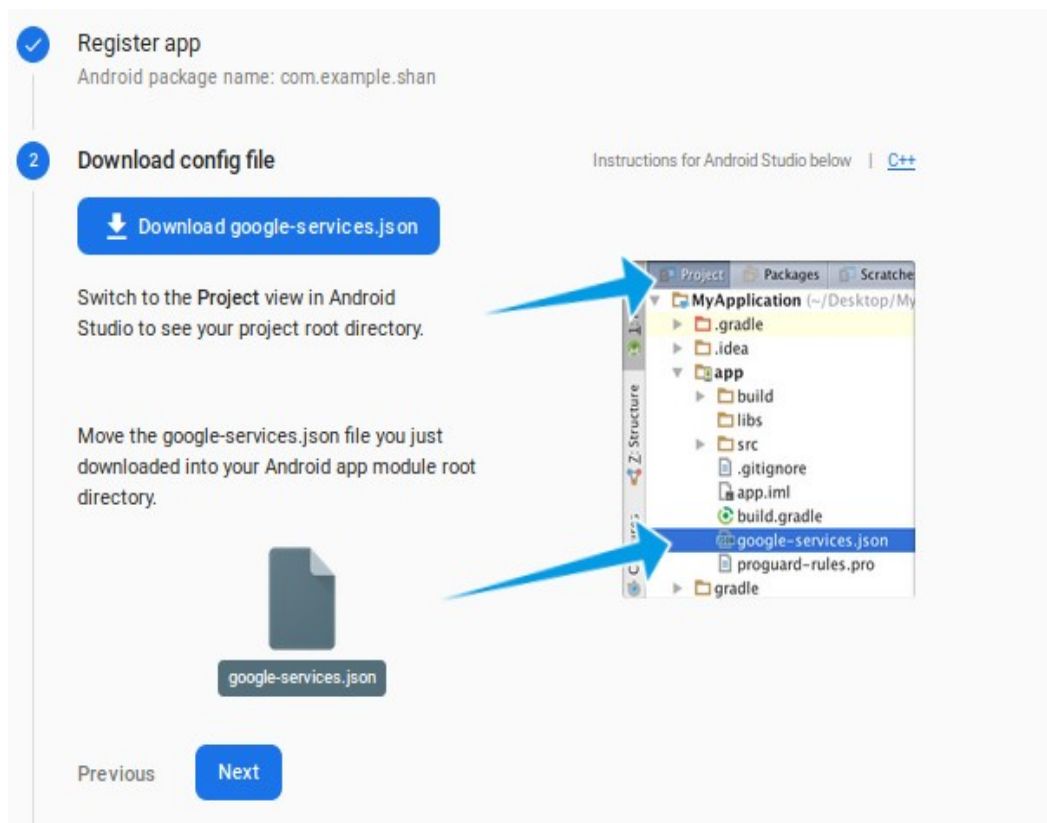
Step 1- App Details:

In the first step, shown in the figure asks for package name. Specify the package name of your app and leave the default fields. Proceed by clicking Register app.

The screenshot shows the "Register app" form in the Firebase console. A blue circle with the number "1" is next to the "Register app" title. The form has three main input fields: "Android package name" with the value "com.example.shan.firbasedemoapp1", "App nickname (optional)", and "Debug signing certificate SHA-1 (optional)". Below these fields, there is a note: "Required for Dynamic Links, Invites, and Google Sign-In or phone number support in Auth. Edit SHA-1s in Settings." At the bottom left of the form is a blue "Next" button.

Step 2- Copy Configuration File:

In this step, you have a button named Download google-services.json. Click to download the JSON configuration file. Add the file to your Android App project in Android Studio as shown in the instructions. To do this, you need to change from Android view to Project view.



The screenshot displays the 'Download config file' step in the Android Studio setup wizard. It features a blue button labeled 'Download google-s-services.js on'. Below the button, instructions state: 'Switch to the Project view in Android Studio to see your project root directory.' and 'Move the google-services.json file you just downloaded into your Android app module root directory.' A file icon labeled 'google-services.json' is shown. To the right, a file explorer window for 'MyApplication' is open, showing the project structure. A blue arrow points to the 'Project' view tab, and another blue arrow points to the 'google-services.json' file in the project root directory. The file explorer shows folders like '.gradle', '.idea', 'app', 'build', 'libs', 'src', and files like '.gitignore', 'app.iml', 'build.gradle', 'google-services.json', 'proguard-rules.pro', and 'gradle'.

Step 3- Add Firebase SDK and sync gradle:

The next step is to add Firebase dependencies to gradle and sync it. The instructions are mentioned on the page as shown in the figure below.

Finally, press "Sync now" in the bar that appears in the IDE. This will take some time. Click next to continue.

The Google services plugin for [Gradle](#) loads the `google-services.json` file you just downloaded. Modify your `build.gradle` files to use the plugin.

Project-level `build.gradle` (<project>/`build.gradle`):

```
buildscript {
    repositories {
        // Check that you have the following line (if not, add it):
        google() // Google's Maven repository
    }
    dependencies {
        ...
        // Add this line
        classpath 'com.google.gms:google-services:4.3.2'
    }
}

allprojects {
    ...
    repositories {
        // Check that you have the following line (if not, add it):
        google() // Google's Maven repository
        ...
    }
}
```

App-level `build.gradle` (<project>/<app-module>/`build.gradle`):

```
apply plugin: 'com.android.application'

dependencies {
    // add SDKs for desired Firebase products
    // https://firebase.google.com/docs/android/setup#available-libra
}
...
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'
```

Here a possible error may occur: No matching client found for package name 'com.example.shan.firebasedemoapp1'

This is because the package name of the Firebase project did not match the package name of the Android App.

Step 4- Finishing:

On the last step, click Continue to console to finish adding the project.

ToDo List App with RecyclerView and Firebase Real-time Database

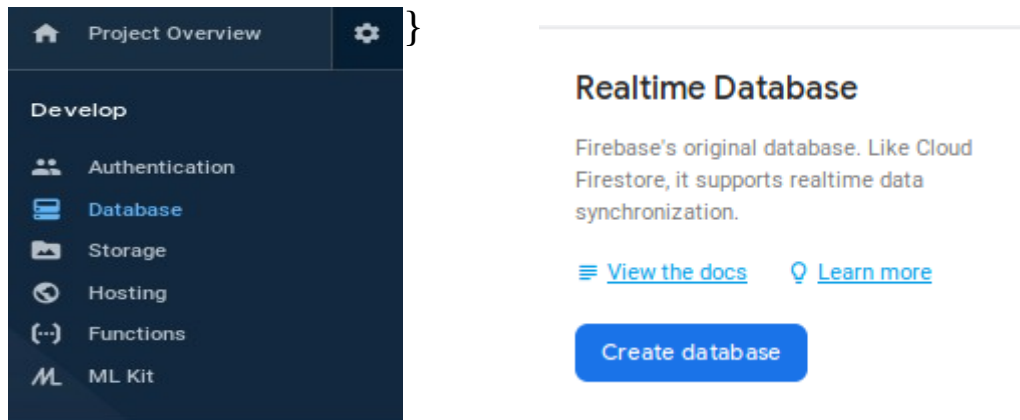
Store and sync data with Firebase NoSQL cloud database. Data is synced across all clients in realtime, and remains available when your app goes offline. The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in realtime to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

We are going to use RecyclerView to list all the task. An EditText widget is used to add new task when a button is clicked. All the task store in Firebase real-time database is retrieved and bind to the RecyclerView through a RecyclerView adapter. A delete icon is used to delete the task from Firebase database.

Setup Firebase Database:

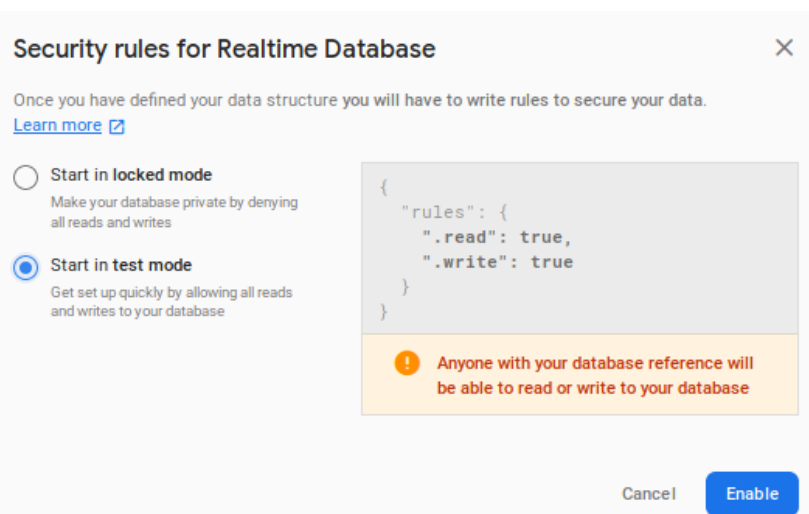
Go to your Firebase Console and click on the database menu link. Click on the RULES menu tab and change the current rules to the code below

```
{  
"rules": {  
  ".read": true,  
  ".write": true  
}
```



Or open the project created. In the left panel click Develop and then Database.

Locate Realtime Database and click Create database. You will see Security rules for Realtime Database. Select Start in test mode and click Enable for now. This is a public access to the database so anybody can get access to your database. Although this is security risk but for testing, it is OK.



Android Client Application:

Below is the screen-shot of the application we will be creating.

Note that the app requires a delete.png icon.

For more icons,

<https://material.io/resources/icons/?style=baseline>



The code is accessible at:

<https://github.com/shanniz/fBaseApp1.git>

Program:

We will create a to do task application with backend of Firebase Real-time database. The main Firebase classes include FirebaseDatabase, DatabaseReference and DataSnapshot.

FirebaseDatabase: Base class usually used to get instance of the database object.

DatabaseReference: The main class used to push and retrieve the Data from Firebase server.

DataSnapshot: A copy of the Firebase database at any given time is represented by DataSnapshot class.

The main interface of the application will have a RecyclerView to display a list of tasks. Each row of the RecyclerView will have TextView to display task and an ImageView to delete the task. At the end, we will have a EditText and a button to create a new task. The UI of the page is:

listing 1 activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <android.support.v7.widget.RecyclerView
        android:id="@+id/task_list"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_above="@+id/add_task_box"
        android:layout_marginBottom="12dp"
        android:orientation="vertical"
    />
    <EditText
        android:id="@+id/add_task_box"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentBottom="true"
        android:inputType="text"
        android:layout_marginStart="86dp"
        android:layout_marginBottom="79dp"
        android:ems="10"
        android:layout_weight="7"
    />
    <Button
        android:id="@+id/add_task_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/add_task_box"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="-79dp"
        android:onClick="addTask"
        android:layout_weight="3"
        android:text="New Task" />
</RelativeLayout>
```

Next we have layout for individual row:

Listing 2: to_do_list.xml

```
<?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="wrap_content"
    android:paddingTop="18dp"
    android:paddingBottom="18dp"
    android:orientation="horizontal">
    <TextView
        android:id="@+id/task_title"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="70"
        android:text="@string/app_name"
        android:textSize="16sp"/>
    <ImageView
        android:id="@+id/task_delete"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:src="@drawable/delete_black_18dp"
        android:layout_weight="15"
        android:contentDescription="@string/app_name" />
</LinearLayout>
```

Now add a class representing task. This simple class contains task

Listing 3: Task.java

```
public class Task {
    private String mTask;
    public Task(String task) {
        this.mTask = task;
    }
    public String getTask() {
        return mTask;
    }
}
```

Create a RecyclerViewAdapter class and RecyclerViewHolder class to render list of tasks in the RecyclerView.

Listing 4: RecyclerViewAdapter.java

```
import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import java.util.List;
```

```

public class RecyclerViewAdapter extends
RecyclerView.Adapter<RecyclerViewHolders> {
    private List<Task> task;
    protected Context context;
    public RecyclerViewAdapter(Context context, List<Task> task) {
        this.task = task;
        this.context = context;
    }
    @Override
    public RecyclerViewHolders onCreateViewHolder(ViewGroup parent, int
viewType) {
        RecyclerViewHolders viewHolder = null;
        View layoutView =
LayoutInflater.from(parent.getContext()).inflate(R.layout.to_do_list,
parent, false);
        viewHolder = new RecyclerViewHolders(layoutView, task);
        return viewHolder;
    }
    @Override
    public void onBindViewHolder(RecyclerViewHolders holder, int
position) {
        holder.taskTitle.setText(task.get(position).getTask());
    }
    @Override
    public int getItemCount() {
        return this.task.size();
    }
}

```

Listing 5: RecyclerViewHolders.java

```

import android.support.v7.widget.RecyclerView;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
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 java.util.List;
public class RecyclerViewHolders extends RecyclerView.ViewHolder{
    private static final String TAG =
RecyclerViewHolders.class.getSimpleName();
    public TextView taskTitle;
    public ImageView deleteIcon;
    private List<Task> taskObject;
}

```



```

    public RecyclerViewHolders(final View itemView, final List<Task>
taskObject) {
        super(itemView);
        this.taskObject = taskObject;
        taskTitle = (TextView)itemView.findViewById(R.id.task_title);
        deleteIcon =
(ImageView)itemView.findViewById(R.id.task_delete);
        deleteIcon.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                String taskTitle =
taskObject.get(getAdapterPosition()).getTask();

                DatabaseReference ref =
FirebaseDatabase.getInstance().getReference();
                Query dQuery =
ref.orderByChild("task").equalTo(taskTitle);
                dQuery.addListenerForSingleValueEvent(new
ValueEventListener() {
                    @Override
                    public void onDataChange(DataSnapshot dataSnapshot) {
                        for (DataSnapshot dSnapshot:
dataSnapshot.getChildren()) {
                            dSnapshot.getRef().removeValue();
                        }
                    }
                    @Override
                    public void onCancelled(DatabaseError databaseError)
{
                        Log.e(TAG, "onCancelled",
databaseError.toException());
                    }
                });
            }
        });
    }
}

```

Finally, we will have the main activity that will connect to Firebase database and display the list of tasks in the recyclerview.

When the Button is clicked, the text content of the EditText is sent to the Firebase database through Push() and setValue() methods of the DatabaseReference class.

The addChildEventListener of the DatabaseReference class is used to monitor data change in the Firebase database.

We have created two methods `getAllTask(DataSnapshot dataSnapshot)` and `taskDeletion(DataSnapshot dataSnapshot)` that will be called when a new task is added or deleted from the Firebase database.

Listing 6: MainActivity.java

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import com.google.firebase.database.ChildEventListener;
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 java.util.ArrayList;
import java.util.List;
public class MainActivity extends AppCompatActivity {
    private static final String TAG =
MainActivity.class.getSimpleName();
    private RecyclerView recyclerView;
    private LinearLayoutManager layoutManager;
    private RecyclerViewAdapter recyclerViewAdapter;
    private EditText addTaskBox;
    private DatabaseReference databaseReference;
    private List<Task> allTask;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        allTask = new ArrayList<Task>();
        databaseReference =
FirebaseDatabase.getInstance().getReference();
        addTaskBox = (EditText)findViewById(R.id.add_task_box);
        recyclerView = (RecyclerView)findViewById(R.id.task_list);
        layoutManager = new LinearLayoutManager(this);
        recyclerView.setLayoutManager(layoutManager);
        databaseReference.addChildEventListener(new
ChildEventListener() {
            @Override
```

```

    public void onChildAdded(DataSnapshot dataSnapshot,
String s) {
        getAllTask(dataSnapshot);
    }
    @Override
    public void onChildChanged(DataSnapshot dataSnapshot,
String s) {
        getAllTask(dataSnapshot);
    }
    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {
        taskDeletion(dataSnapshot);
    }
    @Override
    public void onChildMoved(DataSnapshot dataSnapshot,
String s) {
    }
    @Override
    public void onCancelled(DatabaseError databaseError) {
    }
    });
}
public void addTask(View view){
    String enteredTask = addTaskBox.getText().toString();
    if(TextUtils.isEmpty(enteredTask)){
        Toast.makeText(MainActivity.this, "Enter a task first",
Toast.LENGTH_LONG).show();
        return;
    }
    Task taskObject = new Task(enteredTask);
    databaseReference.push().setValue(taskObject);
    addTaskBox.setText("");
}
private void getAllTask(DataSnapshot dataSnapshot){
    for(DataSnapshot singleSnapshot :
dataSnapshot.getChildren()){
        String taskTitle =
singleSnapshot.getValue(String.class);
        allTask.add(new Task(taskTitle));
        recyclerViewAdapter = new
RecyclerViewAdapter(MainActivity.this, allTask);
        recyclerView.setAdapter(recyclerViewAdapter);
    }
}
private void taskDeletion(DataSnapshot dataSnapshot){
    for(DataSnapshot singleSnapshot :
dataSnapshot.getChildren()) {

```

```
String taskTitle =
singleSnapshot.getValue(String.class);
    for(int i = 0; i < allTask.size(); i++){
        if(allTask.get(i).getTask().equals(taskTitle)){
            allTask.remove(i);
        }
    }
    Log.d(TAG, "Task deleted " + taskTitle);
    recyclerViewAdapter.notifyDataSetChanged();
    recyclerViewAdapter = new
RecyclerViewAdapter(MainActivity.this, allTask);
    recyclerView.setAdapter(recyclerViewAdapter);
}
}
```

Build the program and test the application.