



Firestore

Firestore is a mobile platform from Google offering a number of different features related to cloud services, allowing users to save and retrieve data to be accessed from any device or browser. It is a backend platform for building Web, Android and IOS applications. Firestore offers real time database, different APIs, multiple authentication types and hosting platform. Since this is an introductory practical session, covers the basics of **Real Time Database**.

Firestore Real-time Database

Store and sync data with NoSQL cloud database. Data is stored as JSON and synchronized in real-time to every connected client and remains available when your app goes offline.

Advantages

- The data is real-time – every change will automatically update connected clients.
- Offline – Firestore apps remain responsive even when offline because the Firestore Real-time Database SDK persists your data to disk. Once connectivity is reestablished, the client device receives any change it missed.
- Simple & User Friendly – Firestore offers a simple control dashboard. No need for complicated configuration.

Limitations

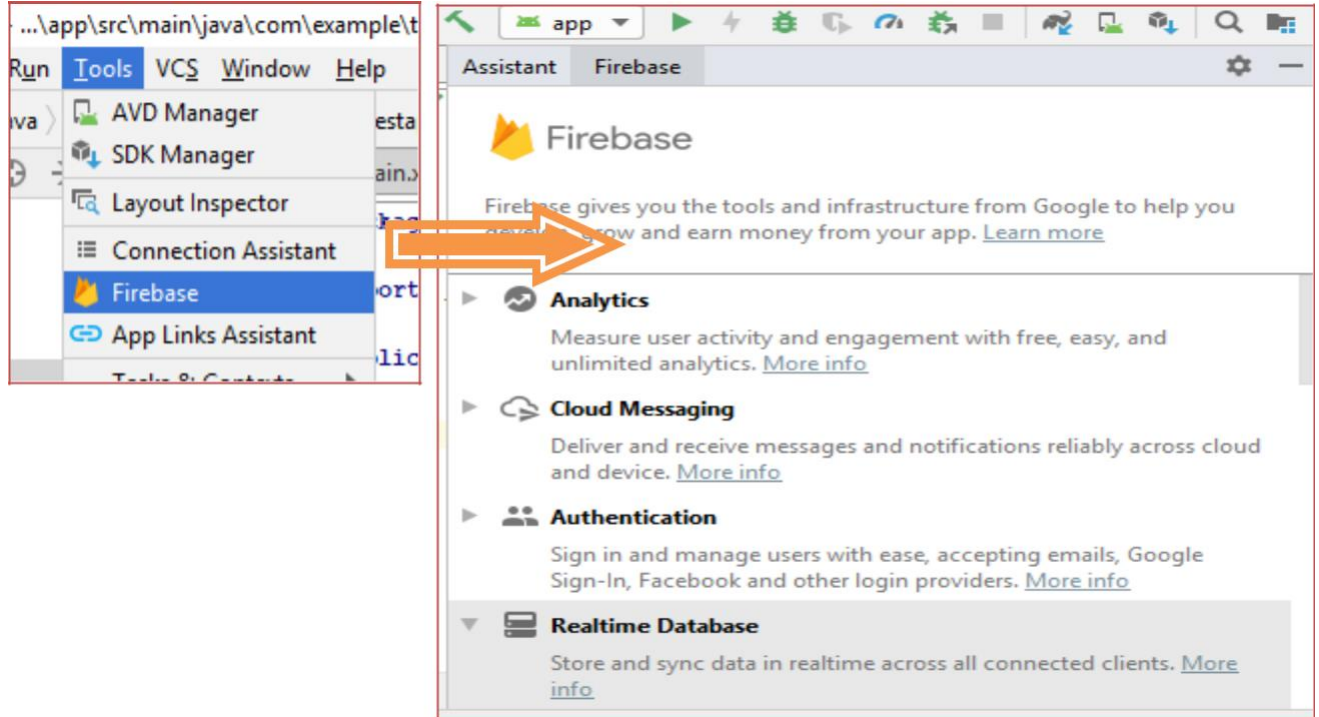
- Firestore free plan is limited to 50 Connections and 100 MB of storage.
- Potential for availability outages – loss of Internet connection, provider outage, provider equipment failure.

Getting Started

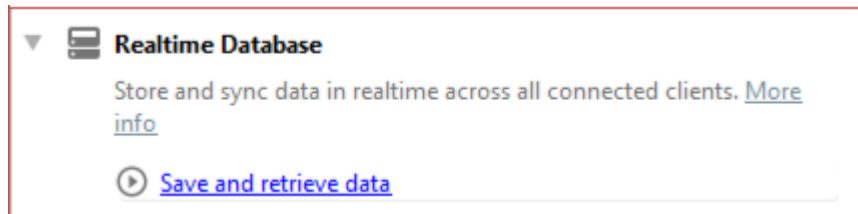
1. Start “Android Studio” and create a new project as “**Test Application – Firestore**”.

Tutorial 06

- Go to **"Tools"** from the main menu of the Android Studio IDE and select **"Firebase"** from there. This will load Firebase pop-up with lot of options.

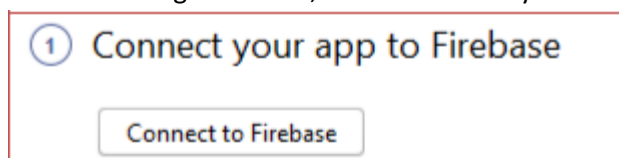


- Click on the right arrow head of the **"Realtime Database"** to expand it and click on the **"Save and retrieve data"** link.



- Start step 1 – Click the button **"Connect to Firebase"**. This will prompt your Google account to login using default browser. Accept all defaults and click on **"Allow"** to allow access.

Note: No Google account, and then create your Google account first...

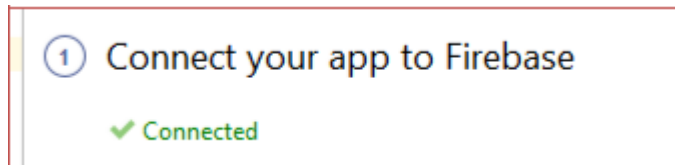


- Check the **"Connect to Firebase"** window. You will see:
 - Login name of your Google account
 - Firebase project name that you are going to create. This name may be the same name as your Android Studio project name.
 Click the **"Connect to Firebase"** button.

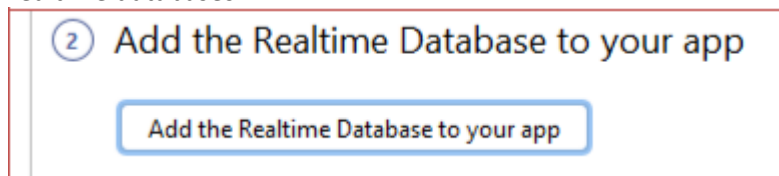
Tutorial 06



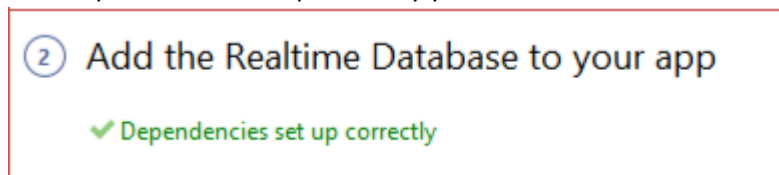
You will see a feedback like below if connected successfully:



6. Start Step 2 – Click the button “Add the Realtime Database to your app” and accept changes. This will make some changes in your Android Studio Project to enable it to work with Firebase realtime databases.

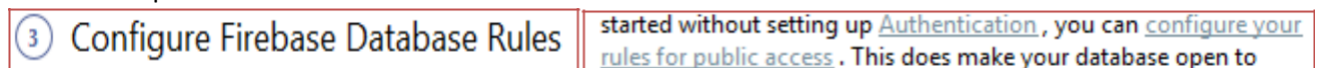


If all dependencies set up correctly you will see a feedback like below:



7. Step 3 – Configure Firebase Database Rules. If you want you can click the link “configure your rules for public access” to get an idea about Database Rules. By default these rules *don't allow* anyone read or write access to your database. Therefore we are going to change these rules to “**PUBLIC**” which means anyone can read or write to your database.

Note: Since this is just a prototype project and we are not going to launch this app it's OK to continue as public...

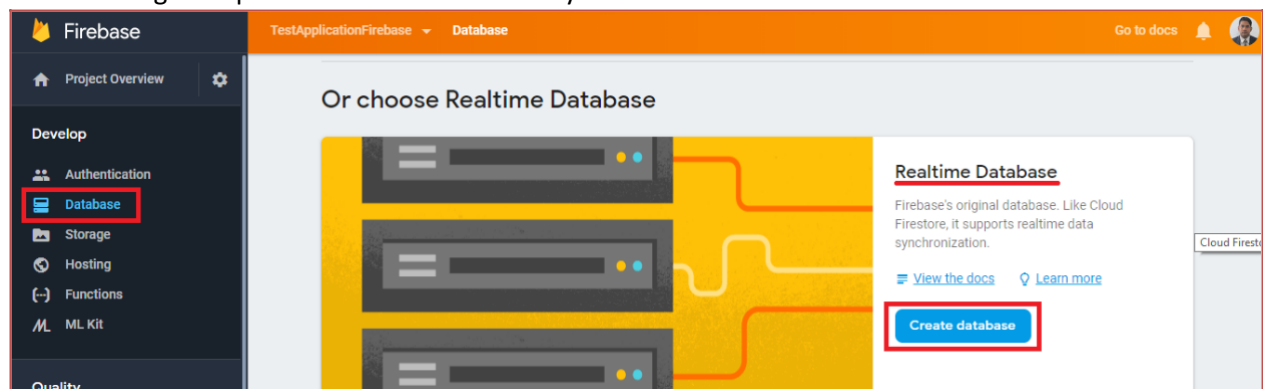


8. Click the link or browse “<https://console.firebase.google.com>” to go the [Firebase Console](https://console.firebase.google.com) and sign in to your Google account to create a Realtime Database and change database rules at the same time. Once you logged in successfully you can see your project as below:

Tutorial 06



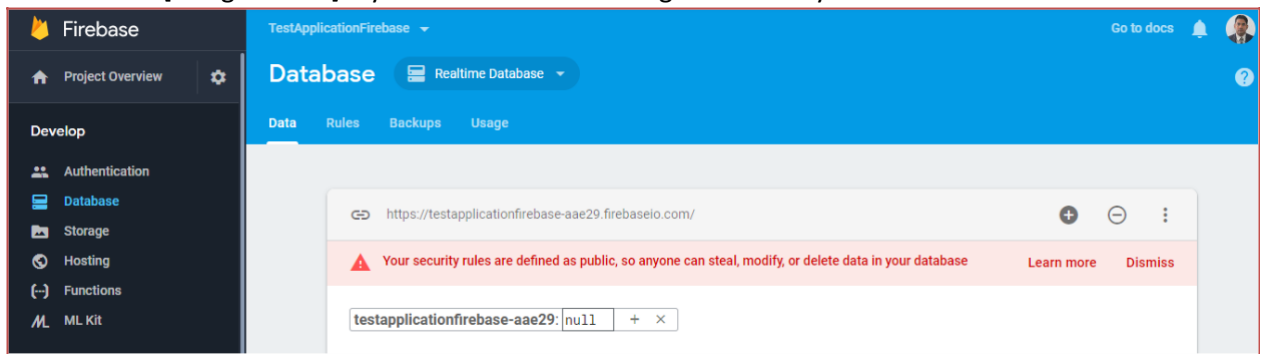
9. Creating a realtime database and changing database rules simultaneously to make it public.
- Click on your Firebase project, Select “**Develop > Database**” from the left hand side navigation panel and scroll down until you see Realtime Database section.



- Click the “**Create database**” button, select “Start in **test mode**” from the pop-up window and click the “**Enable**” button.



- All went well and now you have your realtime database “**testapplicationfirebase**” in there [Google Cloud] if you can see the following instance in your browser.



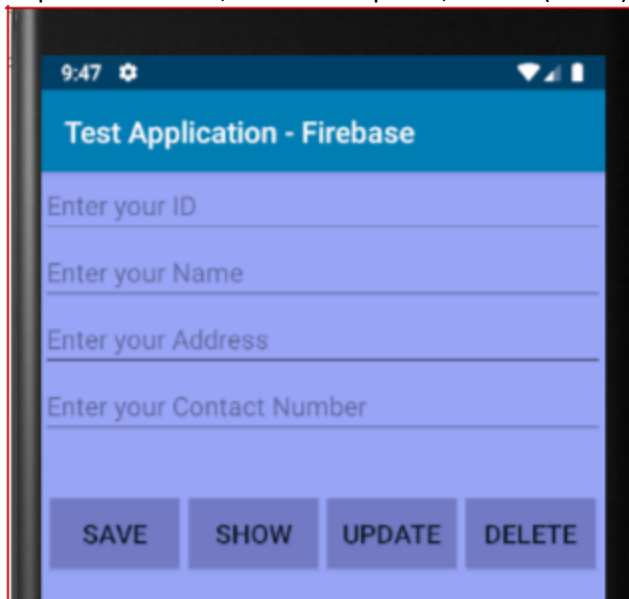
10. Now you are ready to start Step 4:

4 Write to your database

Note: We are going to write a **Student** object to the database with some common attributes as:

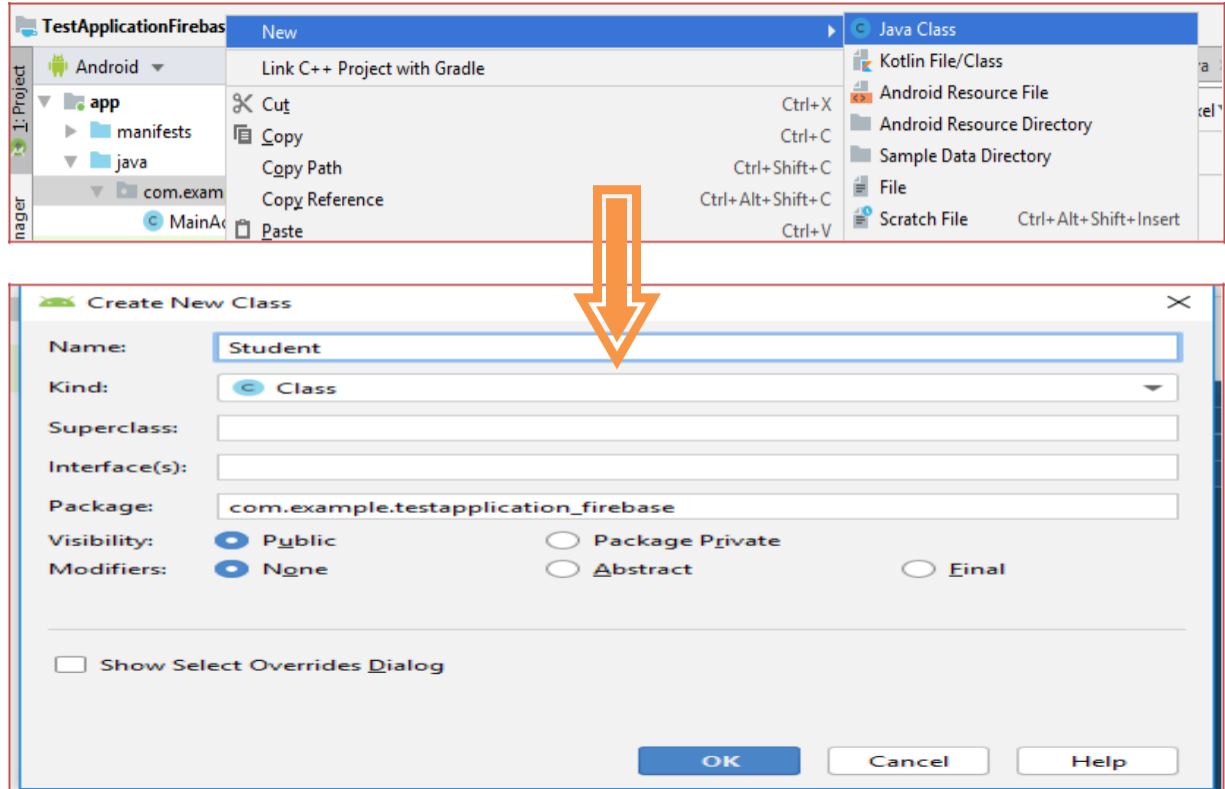
- ID - String
- Name - String
- Address - String
- Contact No - Integer

11. Modify the design of your main activity as follows so that it can take 4 inputs and has 4 buttons to perform Create, Read and Update, Delete (**CRUD**) operations.



Tutorial 06

12. Create a new java class for Student by right click on the package contains Main Activity, give the name as **"Student"** and click the **OK** button.



- Declare Student properties in **Student** class:

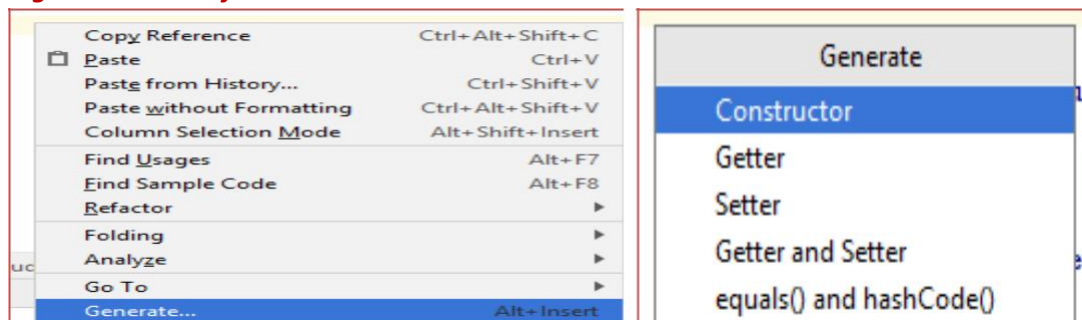
```
package com.example.testapplication_firebase;

public class Student {

    private String ID;
    private String name;
    private String address;
    private Integer conNo;
}
```

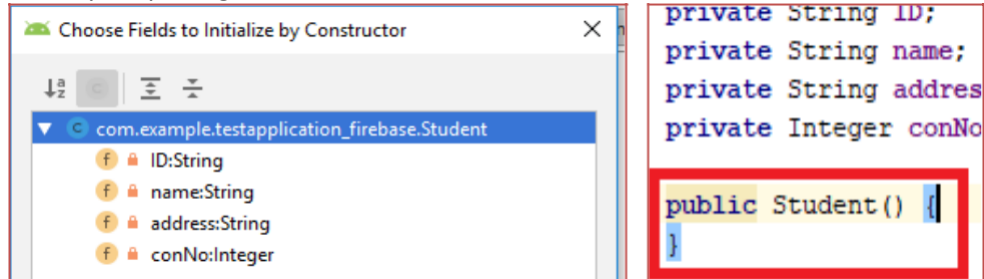
- Add default constructor using the feature provided by this Android Studio IDE:

Right click on the file > Generate > Constructor

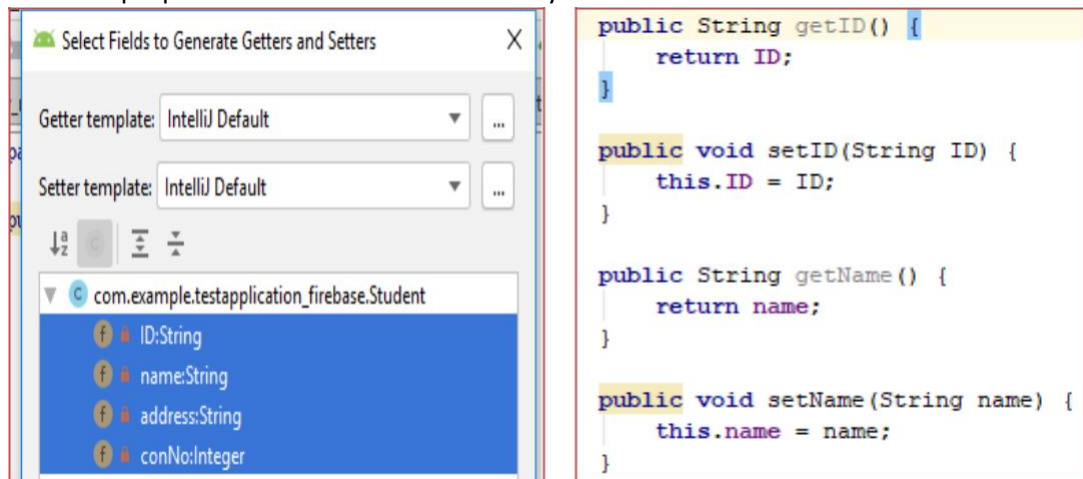


Tutorial 06

- Select your package name and click **OK**:



- Add getter and setter for each property using IDE as above:
Right click on the file > Generate > Getter & Setter
- Select all properties with the aid of **"Shift"** key and click **OK**:



13. Move to **"MainActivity.java"** file and start coding to insert a **Student** object to the database.

- Declare and Define all object type references and a method to clear user inputs:

```
public class MainActivity extends AppCompatActivity {

    EditText txtID, txtName, txtAdd, txtConNo;
    Button btnSave, btnShow, btnUpdate, btnDelete;
    DatabaseReference dbRef;
    Student std;
```

```
//Method to clear all user inputs
private void clearControls() {
    txtID.setText("");
    txtName.setText("");
    txtAdd.setText("");
    txtConNo.setText("");
}
```

Tutorial 06

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    txtID = findViewById(R.id.EtID);
    txtName = findViewById(R.id.EtName);
    txtAdd = findViewById(R.id.EtAddress);
    txtConNo = findViewById(R.id.EtConNo);

    btnSave = findViewById(R.id.BtnSave);
    btnShow = findViewById(R.id.BtnShow);
    btnUpdate = findViewById(R.id.BtnUpdate);
    btnDelete = findViewById(R.id.BtnDelete);

    std = new Student();
```

- Set on click listener to the **"SAVE"** button and start coding to insert:

```
dbRef = FirebaseDatabase.getInstance().getReference().child("Student");
try {
    if (TextUtils.isEmpty(txtID.getText().toString()))
        Toast.makeText(getApplicationContext(), text "Please enter an ID", Toast.LENGTH_SHORT).show();
    else if (TextUtils.isEmpty(txtName.getText().toString()))
        Toast.makeText(getApplicationContext(), text "Please enter a Name", Toast.LENGTH_SHORT).show();
    else if (TextUtils.isEmpty(txtAdd.getText().toString()))
        Toast.makeText(getApplicationContext(), text "Please enter an Address", Toast.LENGTH_SHORT).show();
    else {
        //Take inputs from the user and assigning them to this instance (std) of the Student...
        std.setID(txtID.getText().toString().trim());
        std.setName(txtName.getText().toString().trim());
        std.setAddress(txtAdd.getText().toString().trim());
        std.setConNo(Integer.parseInt(txtConNo.getText().toString().trim()));
        //Insert in to the database...
        dbRef.push().setValue(std);
        //Feedback to the user via a Toast...
        Toast.makeText(getApplicationContext(), text "Data Saved Successfully", Toast.LENGTH_SHORT).show();
        clearControls();
    }
}
catch (NumberFormatException e) {
    Toast.makeText(getApplicationContext(), text "Invalid Contact Number", Toast.LENGTH_SHORT).show();
}
```

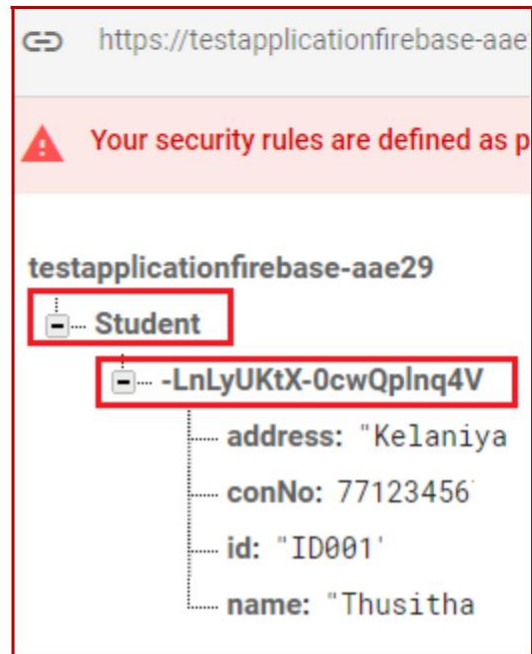

BSc (Hons) in Information Technology



Year 2 IT2010 - Mobile Application Development

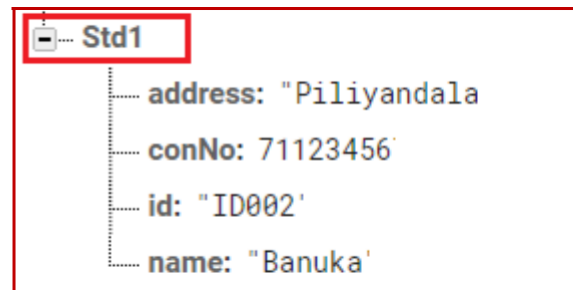
Tutorial 06

14. Run your application and check the database.



15. Add a Student object with your own key. Modify the code, Run your application, Add another student (**Std1**) and check.

```
//Insert in to the database...  
//dbRef.push().setValue(std);  
dbRef.child("Std1").setValue(std);  
//Feedback to the user via a Toast...  
Toast.makeText(getApplicationContext(),  
clearControls());
```



Tutorial 06

16. Retrieve data from database.

- Add a click listener to the **"SHOW"** button and start coding.

```
DatabaseReference readRef = FirebaseDatabase.getInstance().getReference().child("Student").child("Std1");
readRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if (dataSnapshot.hasChildren()) {
            txtID.setText(dataSnapshot.child("id").getValue().toString());
            txtName.setText(dataSnapshot.child("name").getValue().toString());
            txtAdd.setText(dataSnapshot.child("address").getValue().toString());
            txtConNo.setText(dataSnapshot.child("conNo").getValue().toString());
        }
        else
            Toast.makeText(getApplicationContext(), text: "No Source to Display", Toast.LENGTH_SHORT).show()
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {

    }
});
```

- Run your application and click **"SHOW"** button.

Student

Std1

```
address: "Piliyandala"
conNo: 71123456
id: "ID002"
name: "Banuka"
```

Test Application - Firebase

ID002
Banuka
Piliyandala
711234567

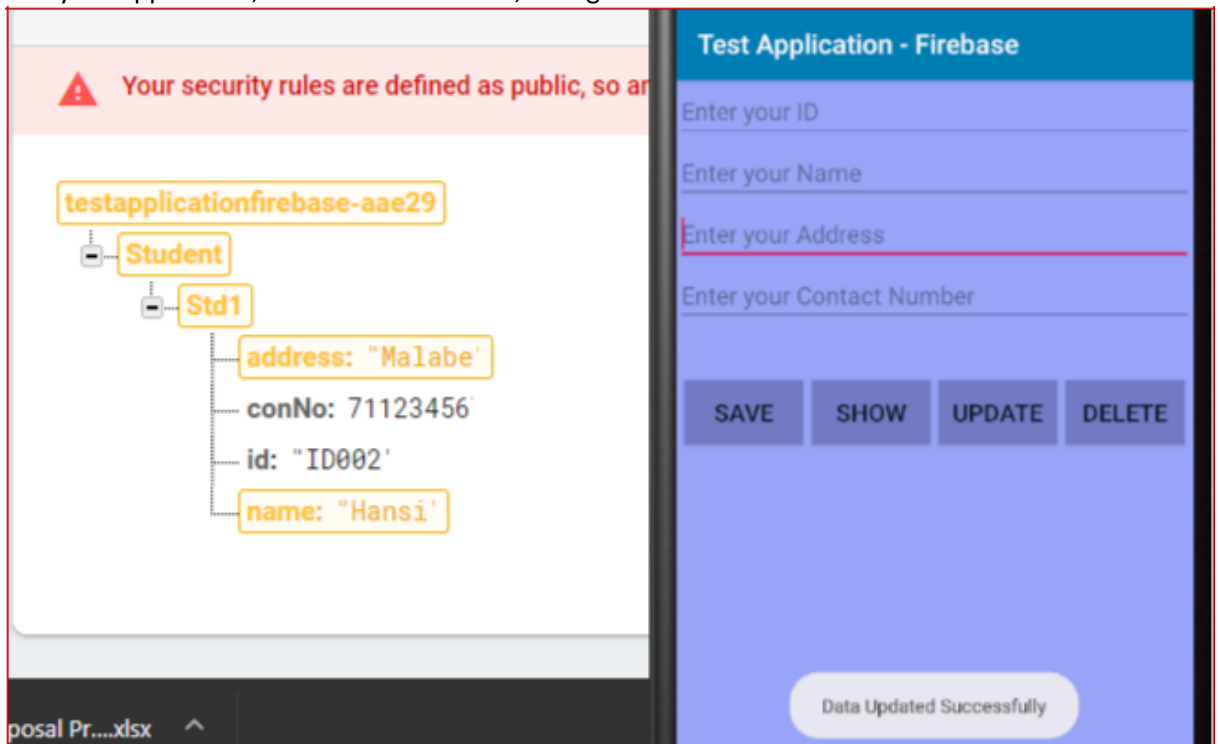
17. Update date.

- Add a click listener to the “**UPDATE**” button and start coding

```
DatabaseReference updRef = FirebaseDatabase.getInstance().getReference().child("Student");
updRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if (dataSnapshot.hasChild( path: "Std1")) {
            try {
                std.setID(txtID.getText().toString().trim());
                std.setName(txtName.getText().toString().trim());
                std.setAddress(txtAdd.getText().toString().trim());
                std.setConNo(Integer.parseInt(txtConNo.getText().toString().trim()));

                dbRef = FirebaseDatabase.getInstance().getReference().child("Student").child("Std1");
                dbRef.setValue(std);
                clearControls();
                //Feedback to the user via a Toast...
                Toast.makeText(getApplicationContext(), text: "Data Updated Successfully", Toast.LENGTH_SHORT).show();
            }
            catch (NumberFormatException e) {
                Toast.makeText(getApplicationContext(), text: "Invalid Contact Number", Toast.LENGTH_SHORT).show();
            }
        }
        else
            Toast.makeText(getApplicationContext(), text: "No Source to Update", Toast.LENGTH_SHORT).show();
    }
});
```

- Run your application, click “**SHOW**” button, change values and click “**UPDATE**” button



The screenshot displays the Firebase console on the left and the application interface on the right. The Firebase console shows the database structure for 'testapplicationfirebase-aae29' with a 'Student' node containing a 'Std1' child. The 'Std1' node has the following data: address: 'Malabe', conNo: 71123456, id: 'ID002', and name: 'Hansi'. The application interface, titled 'Test Application - Firebase', features input fields for 'Enter your ID', 'Enter your Name', 'Enter your Address', and 'Enter your Contact Number'. Below these fields are four buttons: 'SAVE', 'SHOW', 'UPDATE', and 'DELETE'. At the bottom of the interface, a toast message reads 'Data Updated Successfully'.

Tutorial 06

18. Delete data.

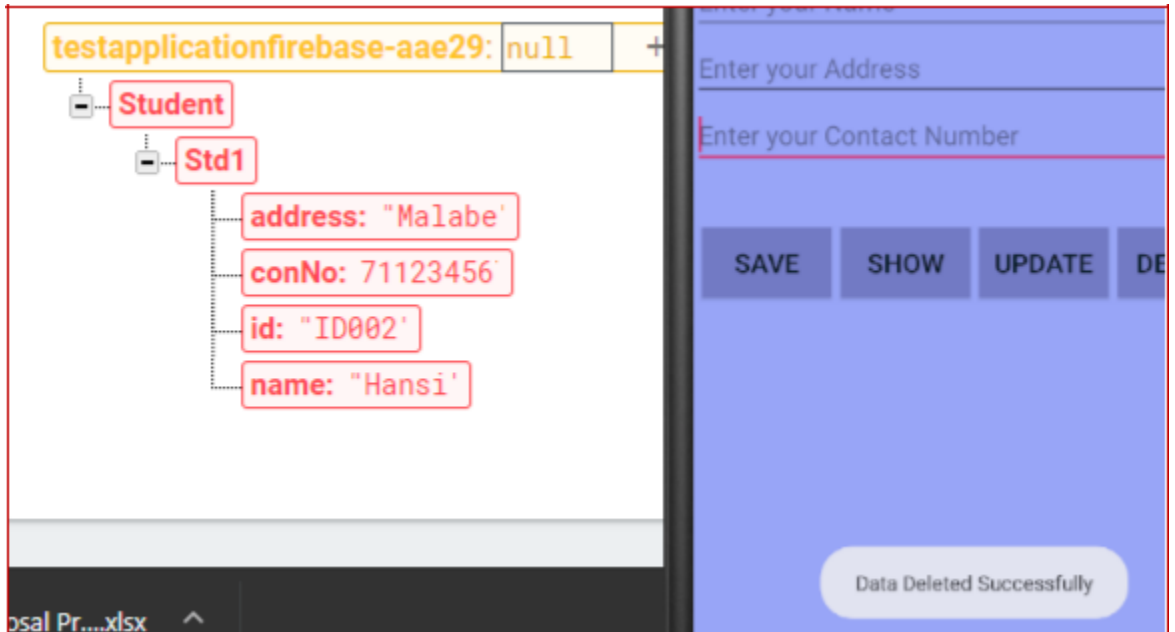
- Add a click listener to the **"DELETE"** button and start coding

```
DatabaseReference delRef = FirebaseDatabase.getInstance().getReference().child("Student");
delRef.addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if (dataSnapshot.hasChild( path: "Std1")) {
            DatabaseReference dbRef = FirebaseDatabase.getInstance().getReference().child("Student").child("Std1");
            dbRef.removeValue();
            clearControls();
            Toast.makeText(getApplicationContext(), text: "Data Deleted Successfully", Toast.LENGTH_SHORT).show();
        }
        else
            Toast.makeText(getApplicationContext(), text: "No Source to Delete", Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {

    }
});
```

- Run your application, click **"SHOW"** button and then click **"DELETE"** button and check the database



The screenshot displays the Firebase Realtime Database structure on the left and the mobile application interface on the right. The database shows a 'Student' node with a child 'Std1' containing the following data:

- address: "Malabe"
- conNo: "71123456"
- id: "ID002"
- name: "Hansi"

The application interface on the right includes input fields for "Enter your Address" and "Enter your Contact Number", buttons for "SAVE", "SHOW", "UPDATE", and "DELETE", and a toast message at the bottom stating "Data Deleted Successfully".

19. Exercise: Modify this app to insert Student objects with auto incrementing ID's...