**What is Firebase?**

A real-time database that saves and updates data in the cloud was created by Google and is called Firebase. On a variety of platforms, including Android, iOS, and the web, we can use Firebase. For the purpose of showing users data, Firebase supports the JSON format. Using the Firebase database platform, a developer may carry out fundamental CRUD (Create, Read, Update, and Delete) activities. **Firebase** is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. Firebase is a product of Google which helps developers to build, manage, and grow their apps easily. It helps developers to build their apps faster and in a more secure way. No programming is required on the firebase side which makes it easy to use its features more efficiently. It provides services to android, ios, web, and unity. It provides cloud storage. It uses NoSQL for the database for the storage of data. Although Firebase works as a database, it also has various other usages, as shown in the illustration below.

1. User Authentication
2. Real-time Database
3. Remote Configuration
4. Analytics
5. Building micro services
6. Game Development

Firebase is Back-end as a service owned by Google which provides server-less back-end to the app developers. It makes app developers to concentrate on front-end by managing back-end itself for developers. You can many features and options like analytics, notifications to apps using FCM, cloud functions, authentication, real-time databases, storage. For testing purposes we can use testlab and crash reporting. Firebase made my development very easy and convince too.

firebase is not an SQL database and it does not stores data in tabular format. It uses JSON tree structure. So for firebase realtime database the structure for above database will be.

**Brief History of Firebase:**

Firebase initially was an online chat service provider to various websites through API and ran with the name Envolve. It got popular as developers used it to exchange application data like a game state in real time across their users more than the chats. This resulted in the separation of the Envolve architecture and it’s chat system. The Envolve architecture was further evolved by it’s founders James Tamplin and Andrew Lee,to what modern day Firebase is in the year 2012.

**Advantages of Firebase**

Firebase comes with many advantages, as it reduces workload and saves costs for a developer. We mainly use this database because of the following benefits:

1. Real-time updating of data
2. Offline availability
3. Centralized database for all connected devices
4. Easy understanding
5. Cloud messaging

**How to connect Firebase with Android Studio**

**Step 1.** Go to Android Studio. If we don’t already have it, we must download and install it first. Create a new project.

**Step 2.** Once our empty project is fully loaded and Gradle files are synced properly, go to tools -> Firebase. Once a new window opens, locate and select Save and retrieve under the Realtime Database section.

**Step 3.** Connect our Firebase project and move to set our dependencies.

Now CRUD operations can be able to done by using Firebase.

**CRUD Operation in Android using Firebase Database.**

The Android SDK that we are going to use here is the set of development tools to develop applications for the android platform. The database we are using is Firebase which is a cloud-hosted database. So, let us start with the development. Crud Stands for **Create, Read, Update and Delete** operation in Database.

### **Getting Database Reference**

* First we need to get the Firebase Database Reference. You can use DatabaseReference to get the reference.

private DatabaseReference mDatabase;

mDatabase = FirebaseDatabase.getInstance().getReference("path"); //Dont pass any path if you want root of the tree

* The data is stored in the JSON Tree form so you need to get the reference of an specified path. Like in the above database we can get all the Artists by passing “Artists”. If you want to access everything don’t pass anything and it will create a reference of the root of the tree.

### **Write Operation**

* **setValue() –** This method will take a model java class object that will hold all the variables to be stored in the reference. The same method will be used to update the values at it overwrites the data of the specified reference.
* Suppose we have to store an Artist to our reference then we will create a model class as below.

public class Artist {

private String artistId;

private String artistName;

private String artistGenre;

public Artist(){

}

public Artist(String artistId, String artistName, String artistGenre) {

this.artistId = artistId;

this.artistName = artistName;

this.artistGenre = artistGenre;

}

public String getArtistId() {

return artistId;

}

public String getArtistName() {

return artistName;

}

public String getArtistGenre() {

return artistGenre;

}

}

* Now to save the artist we will use setValue() method in MainActivity.java.

Artist artist = new Artist(id, name, genre);

databaseReference.child(id).setValue(artist);

* The update operation will also be done in the same way.

### **Read Operation**

* We will attache a **ValueEventListener** to the reference to read the data.

databaseReference.addValueEventListener(new ValueEventListener() {

@Override

public void onDataChange(DataSnapshot dataSnapshot) {

}

@Override

public void onCancelled(DatabaseError databaseError) {

}

});

* Whenever you will change something in the Database the method **onDataChange()** will be executed. It contains all the data inside the specified path in the reference. We can use the **DataSnapshot** object to read all the data inside the reference.  If some error occurres **onCancelled()** method will be called.
* **onDataChange()** method will also called once after the app launch and hence you can read the data at starting as well.

### **Delete Operation**

* **removeValue()**can be used to delete the data.