Firebase

SEG2105 - Introduction to Software Engineering – Fall 2024

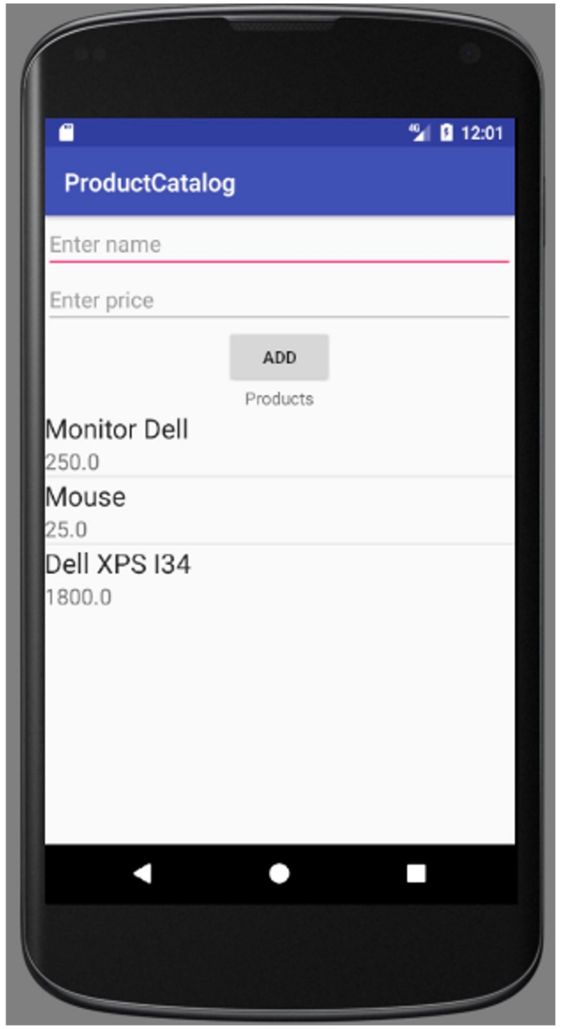
Lab Exercise – Firebase product manager

User Interface

Main Interface:

RelativeLayout

- -> EditText
- -> EditText
- -> Button
- -> TextView
- -> ListView

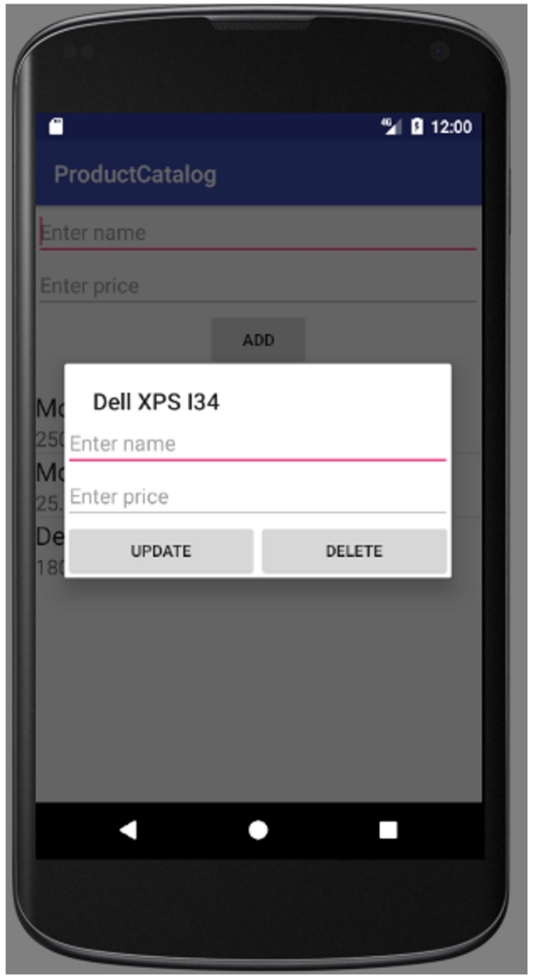


Monitor Dell
250.0
Mouse
25.0
Dell XPS I34
1800.0

Update Dialog:

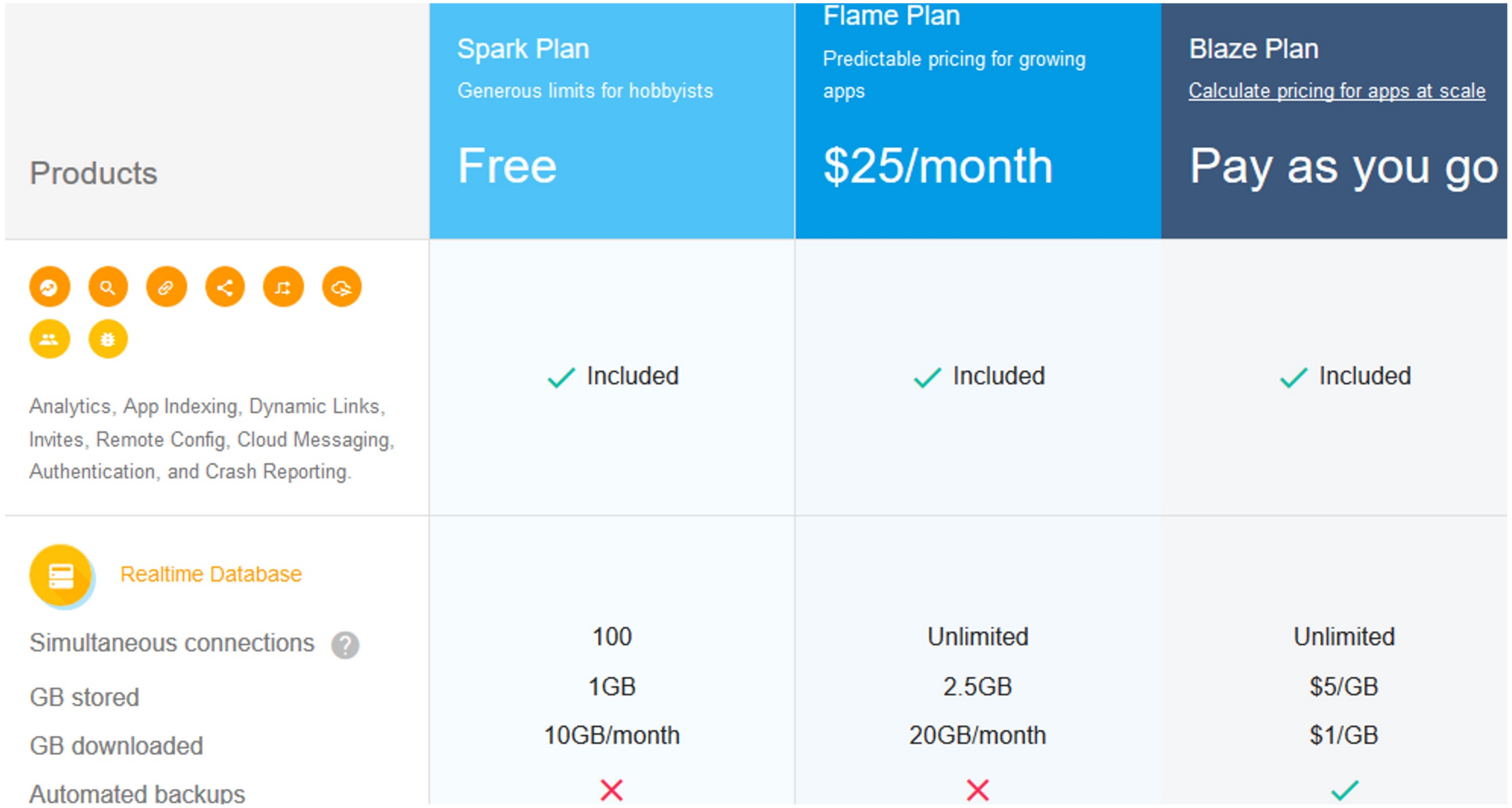
LinearLayout

- -> EditText
- -> EditText
- -> LinearLayout
  - -> Button
  - -> Button



STEP 1 – Make an Account

- The **free account** gives you the ability to have 100 devices connect to Firebase at a single point in time.



Products	Spark Plan Generous limits for hobbyists Free	Flame Plan Predictable pricing for growing apps $25/month	Blaze Plan Calculate pricing for apps at scale Pay as you go
Analytics, App Indexing, Dynamic Links, Invites, Remote Config, Cloud Messaging, Authentication, and Crash Reporting.	✓ Included	✓ Included	✓ Included
Realtime Database			
Simultaneous connections	100	Unlimited	Unlimited
GB stored	1GB	2.5GB	$5/GB
GB downloaded	10GB/month	20GB/month	$1/GB
Automated backups	✗	✗	✓

It's worth noting that 99% of apps never outgrow the free tier, so it's a great tier to start in.

STEP 1 – Make an Account (cont'd)

- The **free account** gives you the ability to have 100 devices connect to Firebase at a single point in time. It's worth noting that 99% of apps never outgrow the free tier, so it's a great tier to start in.

1. Go to https://firebase.google.com and **create an account**.
2. After **logging** in to your account, head over to the [Firebase console](#)
3. And **create a project** that will hold your app's data.

Go to console

Firebase

STEP 2: Connect Firebase to Your Application

- To add Firebase to your android application:

  2a. You can follow a **manual** process (three main steps are required).

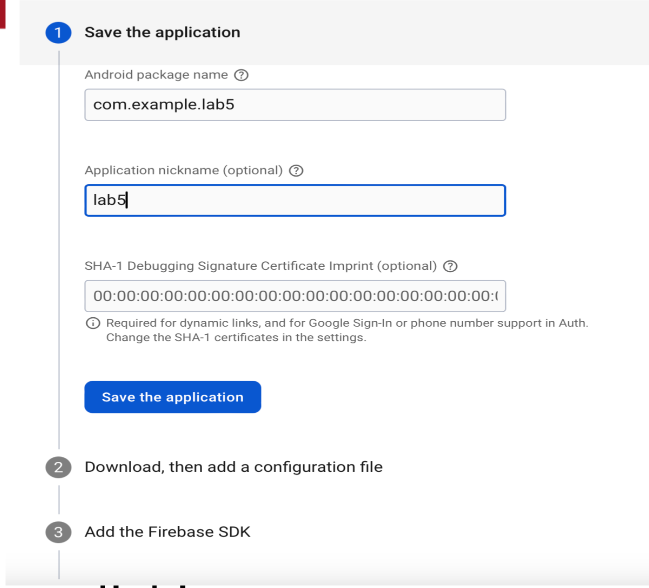
  Or

  2b. You can explore and integrate Firebase services in your app directly from **Android Studio** using the **Assistant** window.

If you're using the latest version of Android Studio (version 2.2 or later), we recommend using the Firebase Assistant to connect your app to Firebase.

- **Prerequisites**
  - A device running Android 15.0 ("VanillaIceCream")or newer, and Google Play services v.49
  - The Google Play services SDK from the **Google Repository**, available in the Android SDK Manager
  - The latest version of Android Studio. **Most recent version is 2024.2.1**

STEP 2a: Manually Connect Firebase to Your Application


Save the application

Android package name: com.example.lab5

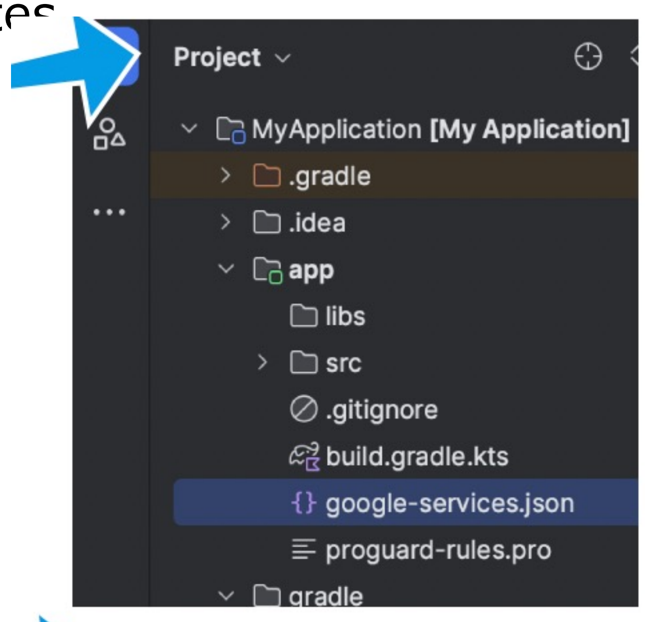
Application nickname (optional): lab5

SHA-1 Debugging Signature Certificate Imprint (optional): 00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:0

Required for dynamic links, and for Google Sign-In or phone number support in Auth. Change the SHA-1 certificates in the settings.

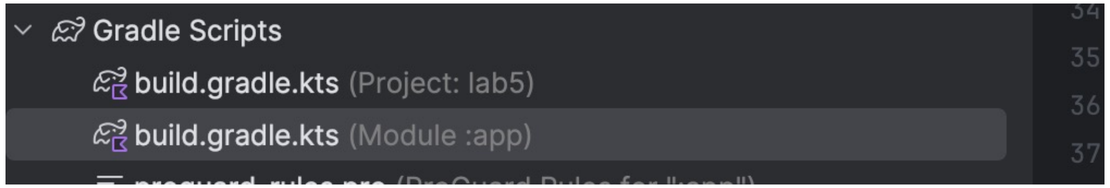
1. **Enter app details:**
   a) **Create a project:** Enter a name and country/region for the project.
   a) **Enter your Android project's package name** in the window after clicking on the Android icon. If your app is going to use certain Google Play services such as Google Sign-In, App Invites and Dynamic Links then you will have to provide the SHA-1 of your signing certificate. The application we will build won't be using any of these services, so leave this field empty. More info about certificates https://developers.google.com/android/guides/client-auth
2. **Copy Config file:** The wizard will allow you to generate a configuration file. **Download** the file (google-services.json) and move it into your Android app module root directory.

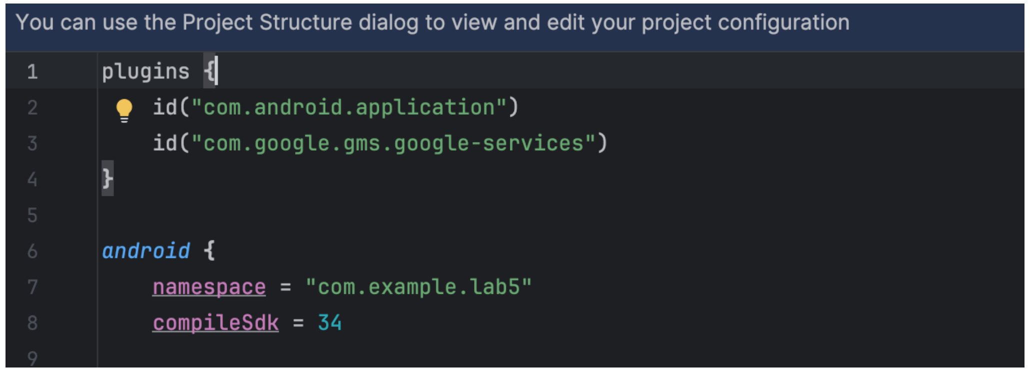
The JSON file contains configuration settings that the Android app needs to communicate with the Firebase servers. It contains details such as the URL to the Firebase project, the API key.



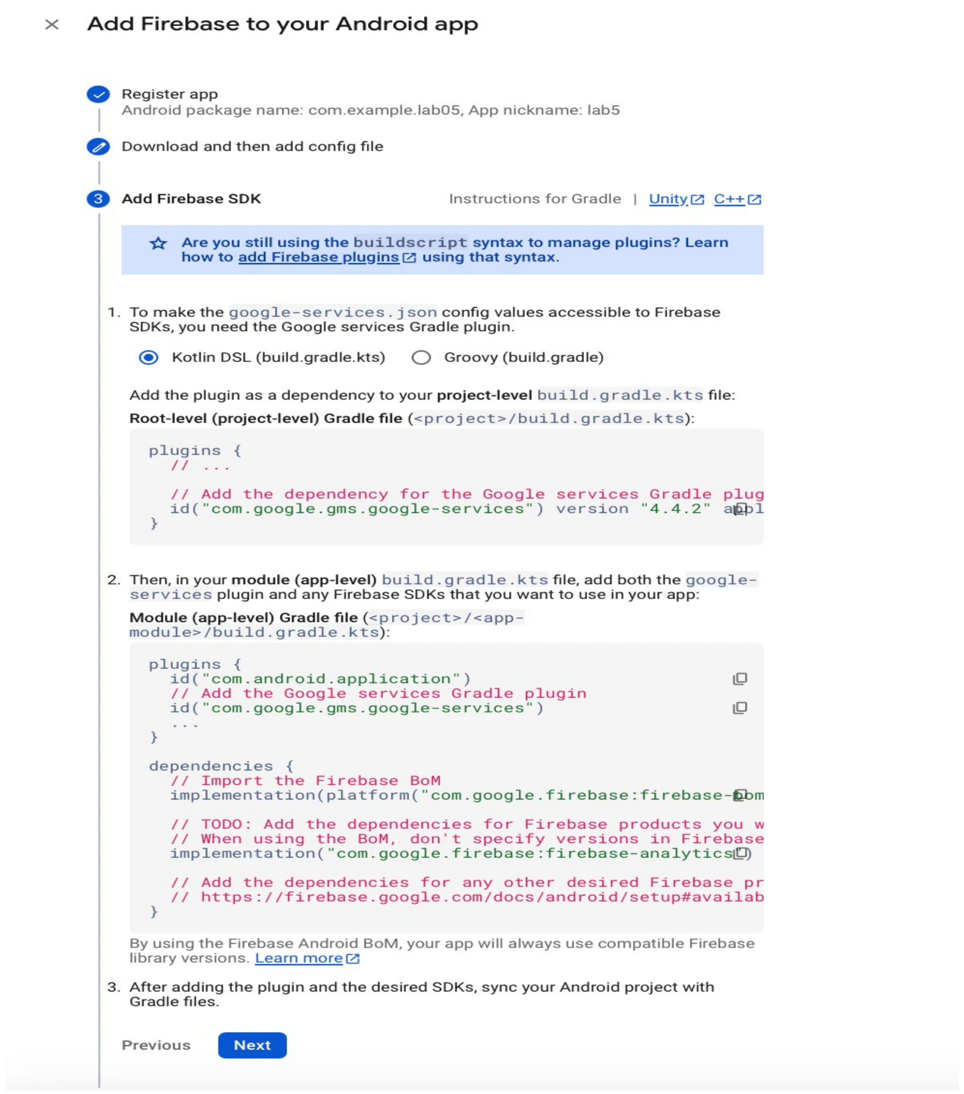
STEP 2a: Manually Connect Firebase to Your Application (cont'd)

3. Add to build.gradle


Gradle Scripts
build.gradle.kts (Project: lab5)
build.gradle.kts (Module :app)
proguard-rules.pro (ProGuard Rules for ":app")


You can use the Project Structure dialog to view and edit your project configuration

```
plugins {
    id("com.android.application")
    id("com.google.gms.google-services")
}

android {
    namespace = "com.example.lab5"
    compileSdk = 34
```


Add Firebase to your Android app

Register app
Android package name: com.example.lab05, App nickname: lab5

Download and then add config file

3 Add Firebase SDK — Instructions for Gradle | Unity C++

Are you still using the `buildscript` syntax to manage plugins? Learn how to add Firebase plugins using that syntax.

1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

   [✓ Kotlin DSL (build.gradle.kts)] Groovy (build.gradle)

   Add the plugin as a dependency to your **project-level** `build.gradle.kts` file:

   Root-level (project-level) Gradle file (`<project>/build.gradle.kts`):

```
plugins {
  // ...

  // Add the dependency for the Google services Gradle plug
  id("com.google.gms.google-services") version "4.4.2" appl
}
```

2. Then, in your **module (app-level)** `build.gradle.kts` file, add both the `google-services` plugin and any Firebase SDKs that you want to use in your app:

   Module (app-level) Gradle file (`<project>/<app-module>/build.gradle.kts`):

```
plugins {
  id("com.android.application")
  // Add the Google services Gradle plugin
  id("com.google.gms.google-services")
  ...
}

dependencies {
  // Import the Firebase BoM
  implementation(platform("com.google.firebase:firebase-bom

  // TODO: Add the dependencies for Firebase products you w
  // When using the BoM, don't specify versions in Firebase
  implementation("com.google.firebase:firebase-analytics")

  // Add the dependencies for any other desired Firebase pr
  // https://firebase.google.com/docs/android/setup#availab
}
```

   By using the Firebase Android BoM, your app will always use compatible Firebase library versions. Learn more

3. After adding the plugin and the desired SDKs, sync your Android project with Gradle files.

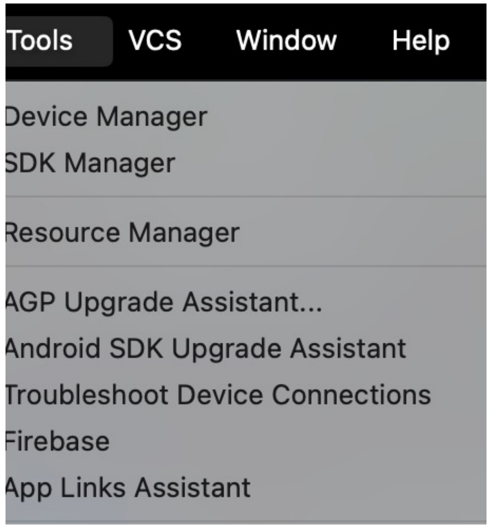
STEP 2b: Connect Firebase to Your Application using Android Studio

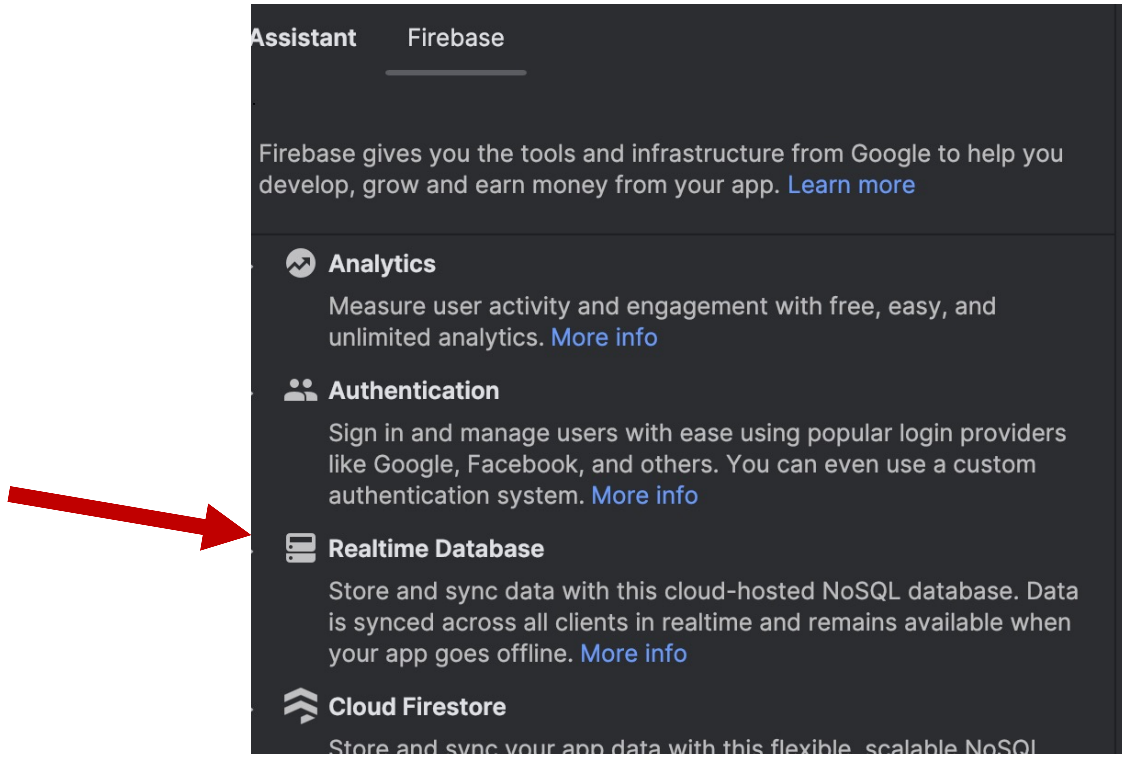
First make sure you have installed Google Repository version 26 or higher, using the following steps:

- Click **Tools > SDK Manager**.
- Click the **SDK Tools** tab.
- Check the **Google Repository** checkbox, and click **OK**.
- Click **OK** to install.
- Click **Background** to complete the installation in the background, or wait for the installation to complete and click **Finish**.

STEP 2b: Connect Firebase to Your Application using Android Studio (cont'd)

1. Click **Tools > Firebase** to open the **Assistant** window.
2. Click to expand one of the listed features then click the Get started with Realtime (under Realtime Database) to connect to Firebase and automatically add the necessary code to your app.





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

Authentication
Sign in and manage users with ease using popular login providers like Google, Facebook, and others. You can even use a custom authentication system. More info

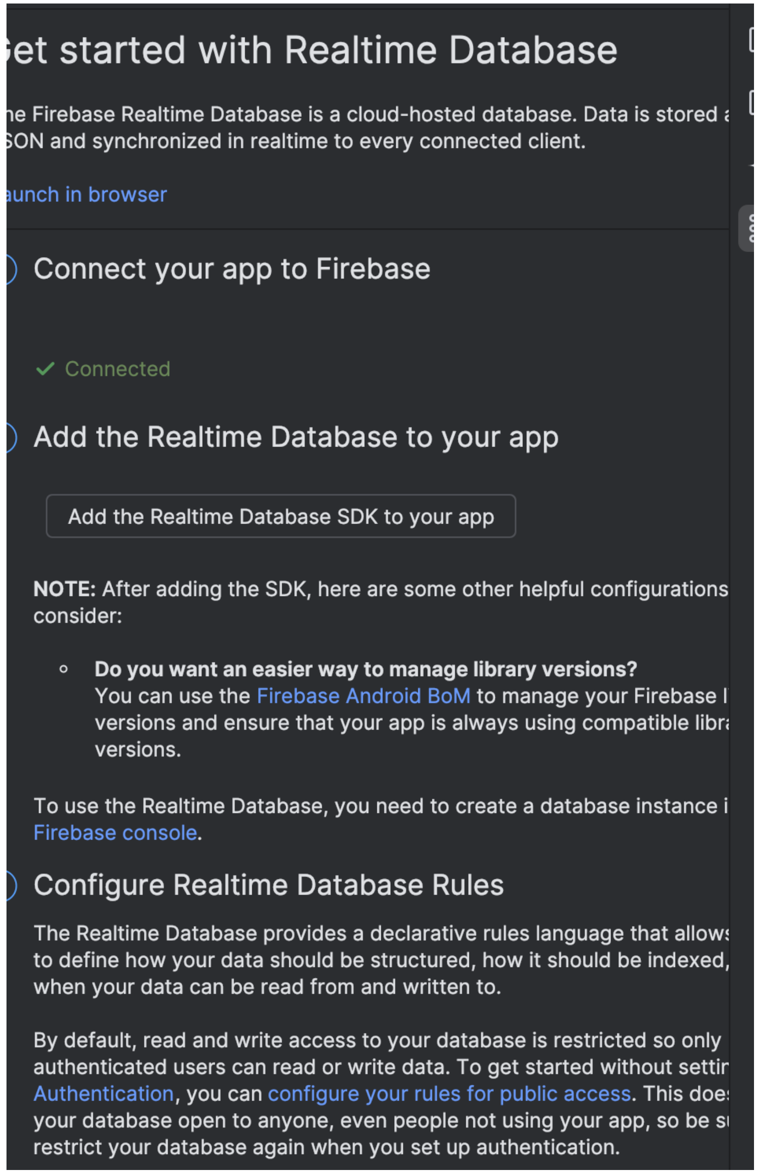
Realtime Database
Store and sync data with this cloud-hosted NoSQL database. Data is synced across all clients in realtime and remains available when your app goes offline. More info

Cloud Firestore
Store and sync your app data with this flexible, scalable NoSQL

STEP 2b: Connect Firebase to Your Application using Android Studio (cont'd)

3. Click **Connect to Firebase** to open the window. Enter the name for your Firebase project.
4. Click **Add the Realtime Database to your app.**

You are now ready to write and read form your Database!



et started with Realtime Database

he Firebase Realtime Database is a cloud-hosted database. Data is stored a
SON and synchronized in realtime to every connected client.

aunch in browser

Connect your app to Firebase

✓ Connected

Add the Realtime Database to your app

Add the Realtime Database SDK to your app

NOTE: After adding the SDK, here are some other helpful configurations
consider:

- **Do you want an easier way to manage library versions?**
  You can use the Firebase Android BoM to manage your Firebase l
  versions and ensure that your app is always using compatible libra
  versions.

To use the Realtime Database, you need to create a database instance i
Firebase console.

Configure Realtime Database Rules

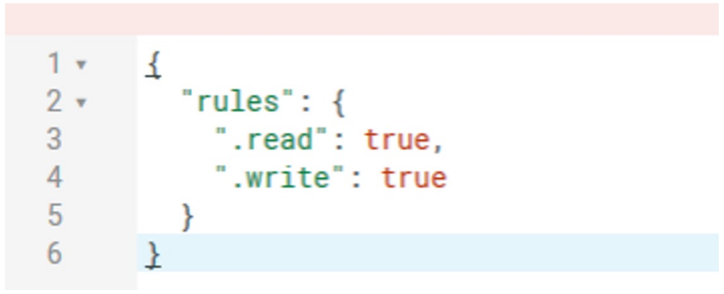
The Realtime Database provides a declarative rules language that allows
to define how your data should be structured, how it should be indexed,
when your data can be read from and written to.

By default, read and write access to your database is restricted so only
authenticated users can read or write data. To get started without settir
Authentication, you can configure your rules for public access. This does
your database open to anyone, even people not using your app, so be su
restrict your database again when you set up authentication.

STEP 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 indexed, and when your data can be read from and written to. **By default, read and write access to your database is restricted so only authenticated users can read or write data.**
- To get started without setting up Authentication , you can configure your rules for public access.

Go to the firebase console and specify your rules as follows:



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

Your turn!

Write and Read Data with Firebase

To download and set up the sample application in Android Studio:

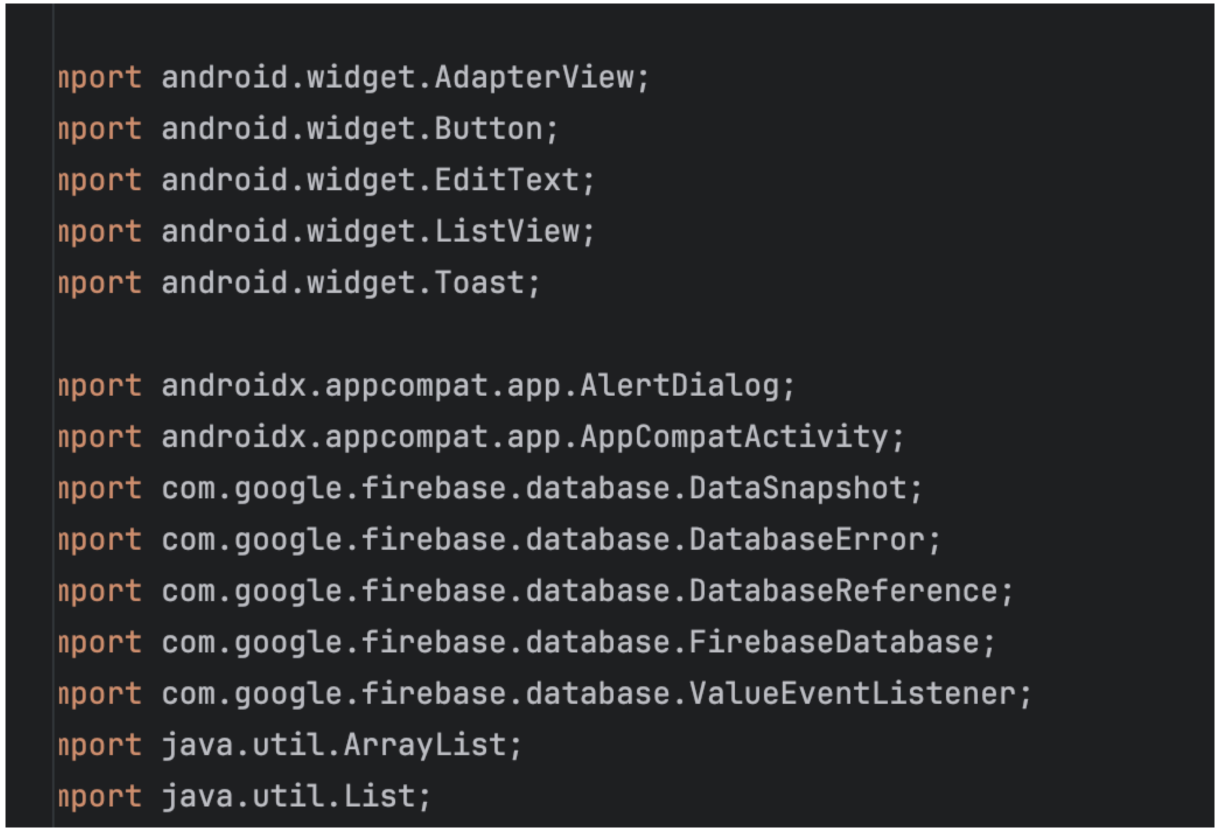
1. **Download** the ProductCatalog sample app from my personal Github.
2. You can either use the "Download ZIP" button on the Github Page or clone on the command line: https://github.com/ri205/lab5

   git clone https://github.com/ri205/lab5.git

3. **Import** the project in Android Studio: Click File > New > Import Project.
4. **Implement** the code to **add**, **delete** and **update** a *product* which will be stored at the Firebase database.

Import Required Libraries

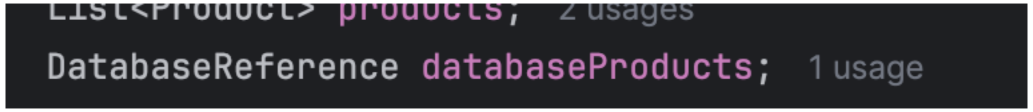
- Import libraries to use Text Boxes, Spinners, Buttons and required Database libraries such as FirebaseDatabase.



```
nport android.widget.AdapterView;
nport android.widget.Button;
nport android.widget.EditText;
nport android.widget.ListView;
nport android.widget.Toast;

nport androidx.appcompat.app.AlertDialog;
nport androidx.appcompat.app.AppCompatActivity;
nport com.google.firebase.database.DataSnapshot;
nport com.google.firebase.database.DatabaseError;
nport com.google.firebase.database.DatabaseReference;
nport com.google.firebase.database.FirebaseDatabase;
nport com.google.firebase.database.ValueEventListener;
nport java.util.ArrayList;
nport java.util.List;
```

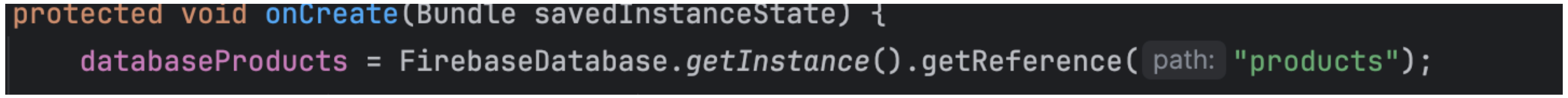
Steps to Read and Write Data on Android

- Get a DatabaseReference
  - To read or write data from the database, you need an instance of DatabaseReference:



```
List<Product> products;  2 usages
DatabaseReference databaseProducts;  1 usage
```

  - Add the following line to the OnCreate method.

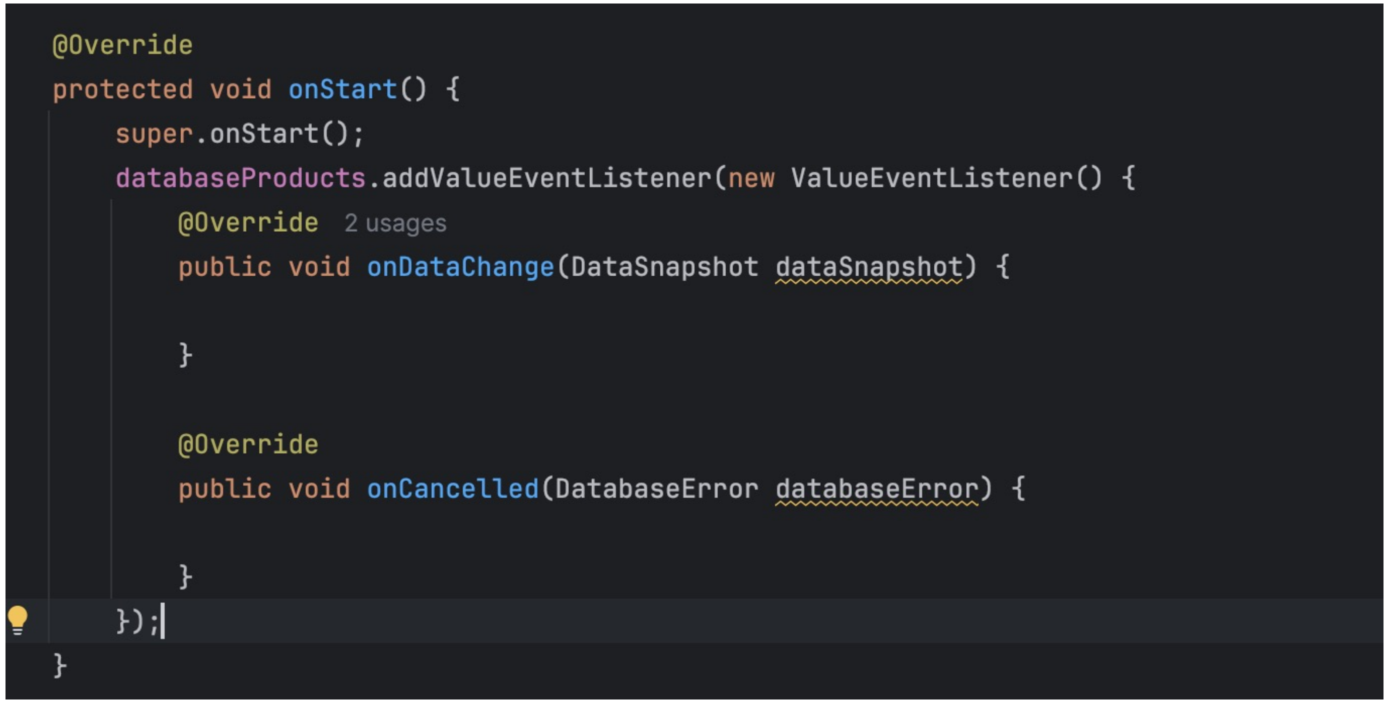


```
protected void onCreate(Bundle savedInstanceState) {
    databaseProducts = FirebaseDatabase.getInstance().getReference( path: "products");
```

[Find the full documentation about reading and writing data on Android here: https://firebase.google.com/docs/database/android/read-and-write](https://firebase.google.com/docs/database/android/read-and-write)

Listen for Value Events

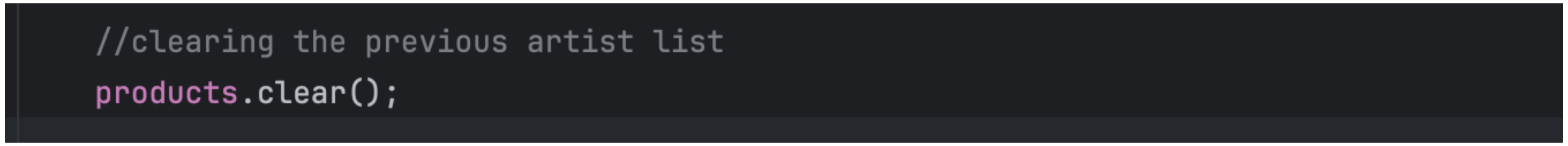
- To read data at a path and listen for changes, use the addValueEventListener() or addListenerForSingleValueEvent() method to add a ValueEventListener to a DatabaseReference.



```
@Override
protected void onStart() {
    super.onStart();
    databaseProducts.addValueEventListener(new ValueEventListener() {
        @Override  2 usages
        public void onDataChange(DataSnapshot dataSnapshot) {

        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });
}
```

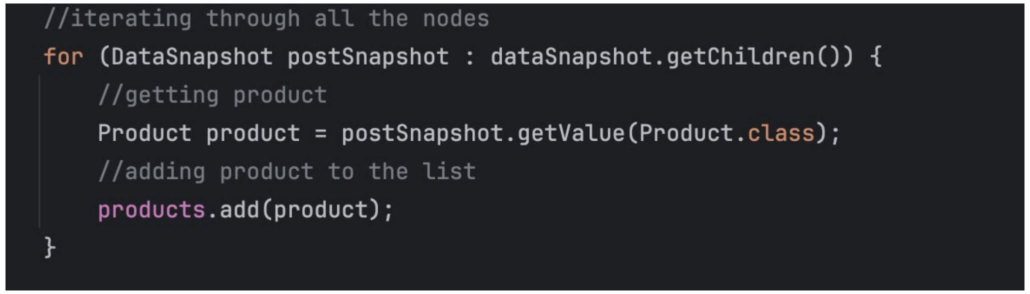
Listen for Value Events (cont'd).

- Add the following to the onDataChange;
  - Clear the previous artis list.



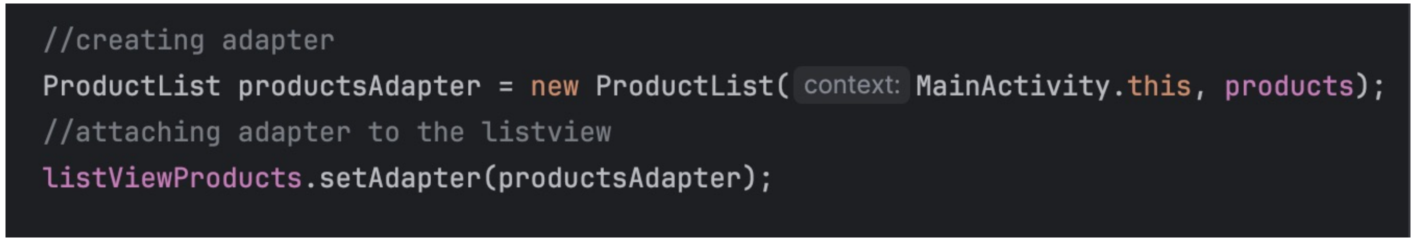
```
//clearing the previous artist list
products.clear();
```

  - Iterate through with the following.



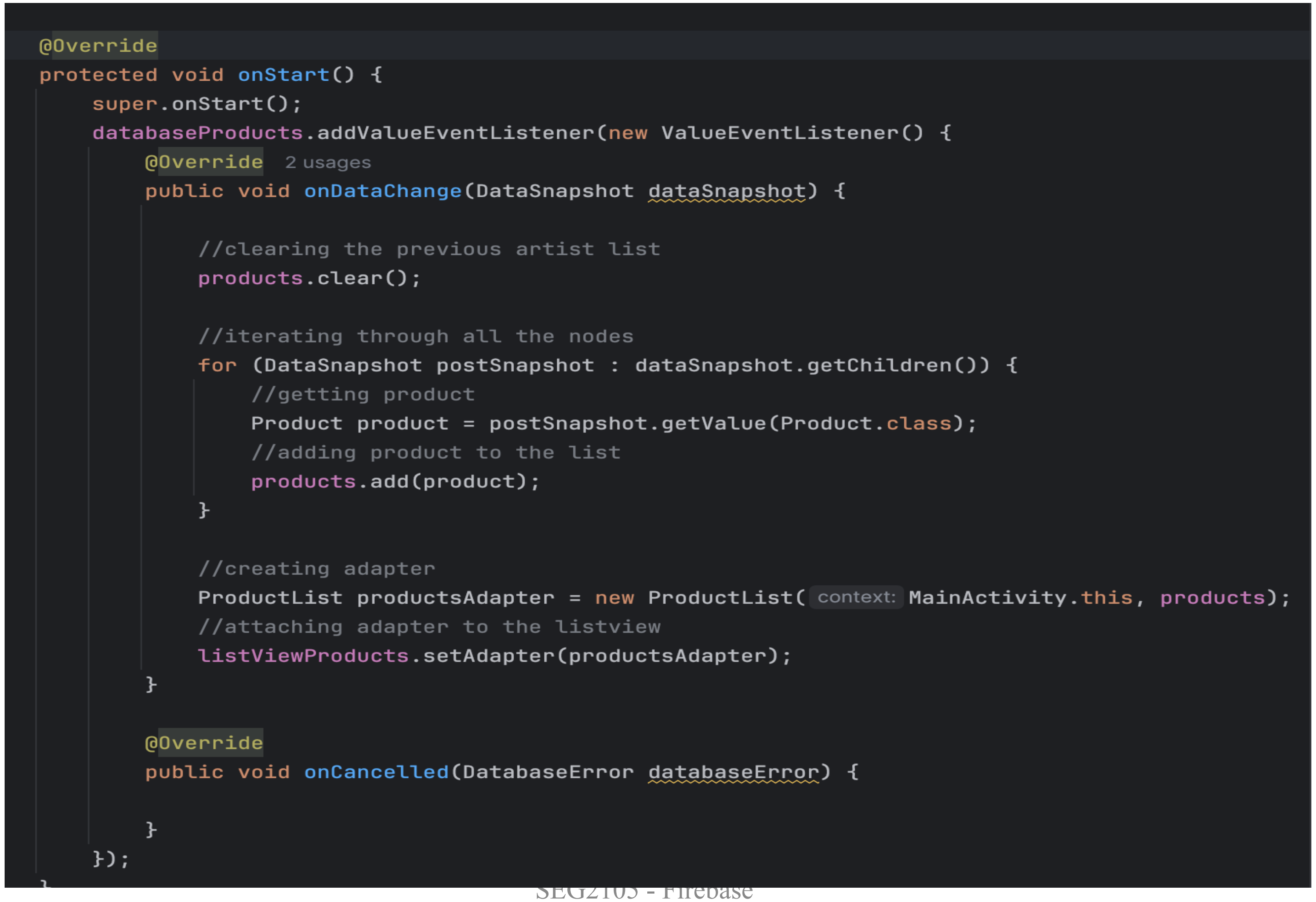
```
//iterating through all the nodes
for (DataSnapshot postSnapshot : dataSnapshot.getChildren()) {
    //getting product
    Product product = postSnapshot.getValue(Product.class);
    //adding product to the list
    products.add(product);
}
```

  - Finally, create the adapter.



```
//creating adapter
ProductList productsAdapter = new ProductList( context: MainActivity.this, products);
//attaching adapter to the listview
listViewProducts.setAdapter(productsAdapter);
```

Listen for Value Events (final look)



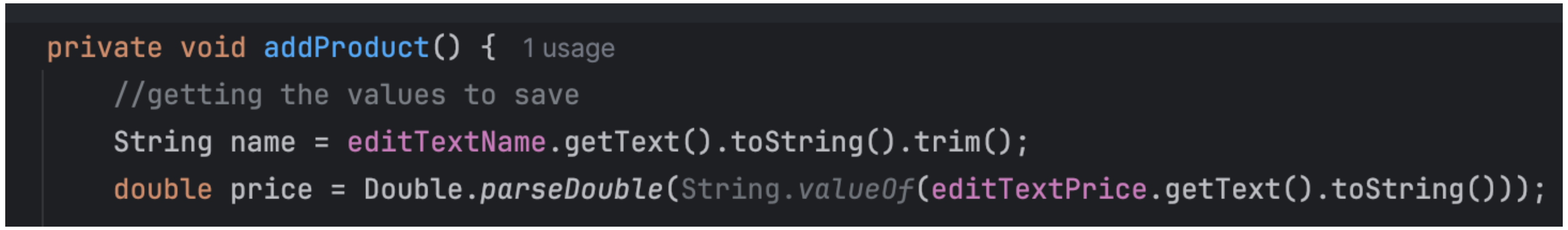
```
@Override
protected void onStart() {
    super.onStart();
    databaseProducts.addValueEventListener(new ValueEventListener() {
        @Override  2 usages
        public void onDataChange(DataSnapshot dataSnapshot) {

            //clearing the previous artist list
            products.clear();

            //iterating through all the nodes
            for (DataSnapshot postSnapshot : dataSnapshot.getChildren()) {
                //getting product
                Product product = postSnapshot.getValue(Product.class);
                //adding product to the list
                products.add(product);
            }

            //creating adapter
            ProductList productsAdapter = new ProductList( context: MainActivity.this, products);
            //attaching adapter to the listview
            listViewProducts.setAdapter(productsAdapter);
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {

        }
    });
```

Read and Write Data

- Basic write operations
- For basic write operations, you can use setValue() to save data to a specified reference, replacing any existing data at that path. You can use this method to:
- Pass types that correspond to the available JSON types as follows:
  - String
  - Long
  - Double
  - Boolean
  - Map<String, Object>
  - List<Object>

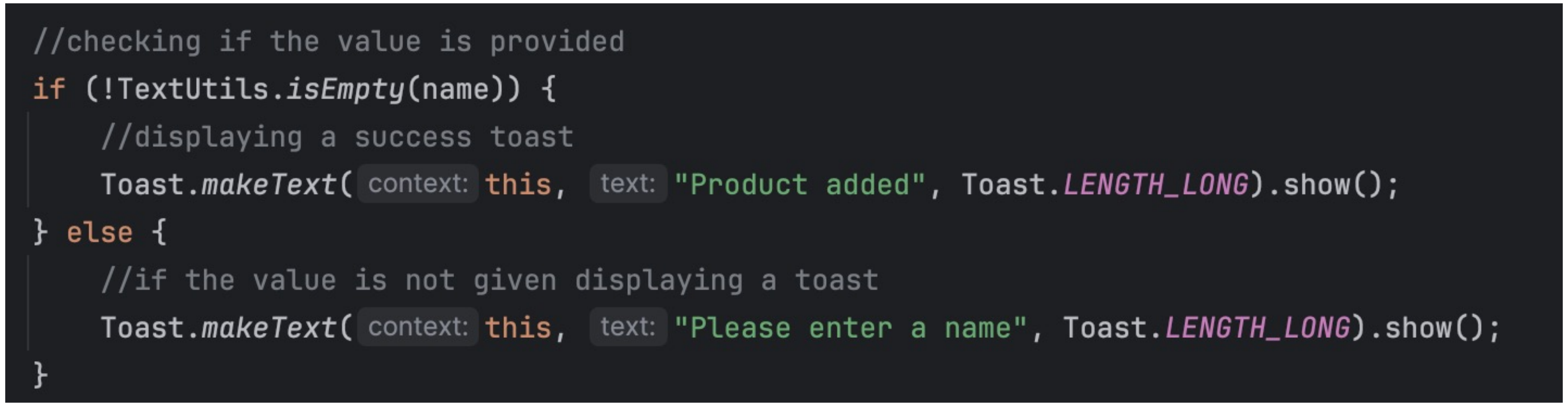
Add a Product to the Database

- Get the values from the TextBoxes for name and the price.



```
private void addProduct() {  1 usage
    //getting the values to save
    String name = editTextName.getText().toString().trim();
    double price = Double.parseDouble(String.valueOf(editTextPrice.getText().toString()));
```

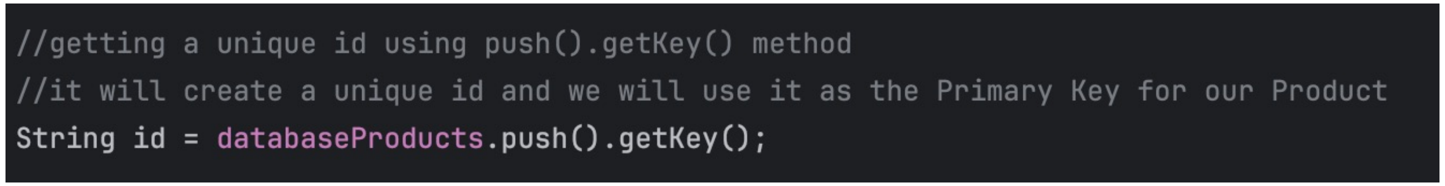
- Inside the addProduct, check the boxes if the values are provided. If the textboxes contain values, do the following and print a success message.



```
//checking if the value is provided
if (!TextUtils.isEmpty(name)) {
    //displaying a success toast
    Toast.makeText( context: this,  text: "Product added", Toast.LENGTH_LONG).show();
} else {
    //if the value is not given displaying a toast
    Toast.makeText( context: this,  text: "Please enter a name", Toast.LENGTH_LONG).show();
}
```

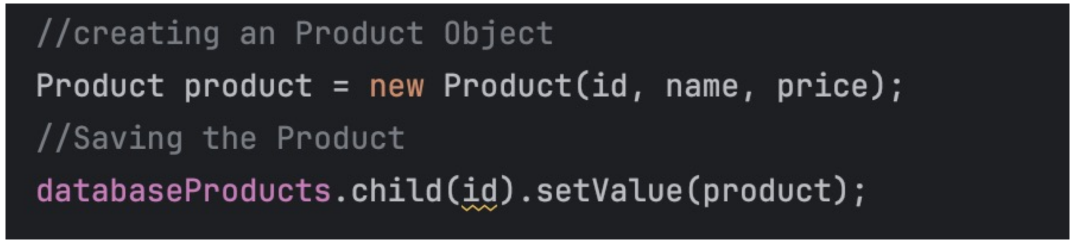
Add a Product to the Database (cont’d).

- Get a unique id for the each product to be saved to the database.



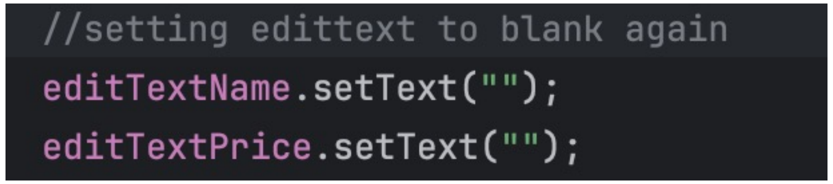
```java
//getting a unique id using push().getKey() method
//it will create a unique id and we will use it as the Primary Key for our Product
String id = databaseProducts.push().getKey();
```

- Create a Product object and save this object.



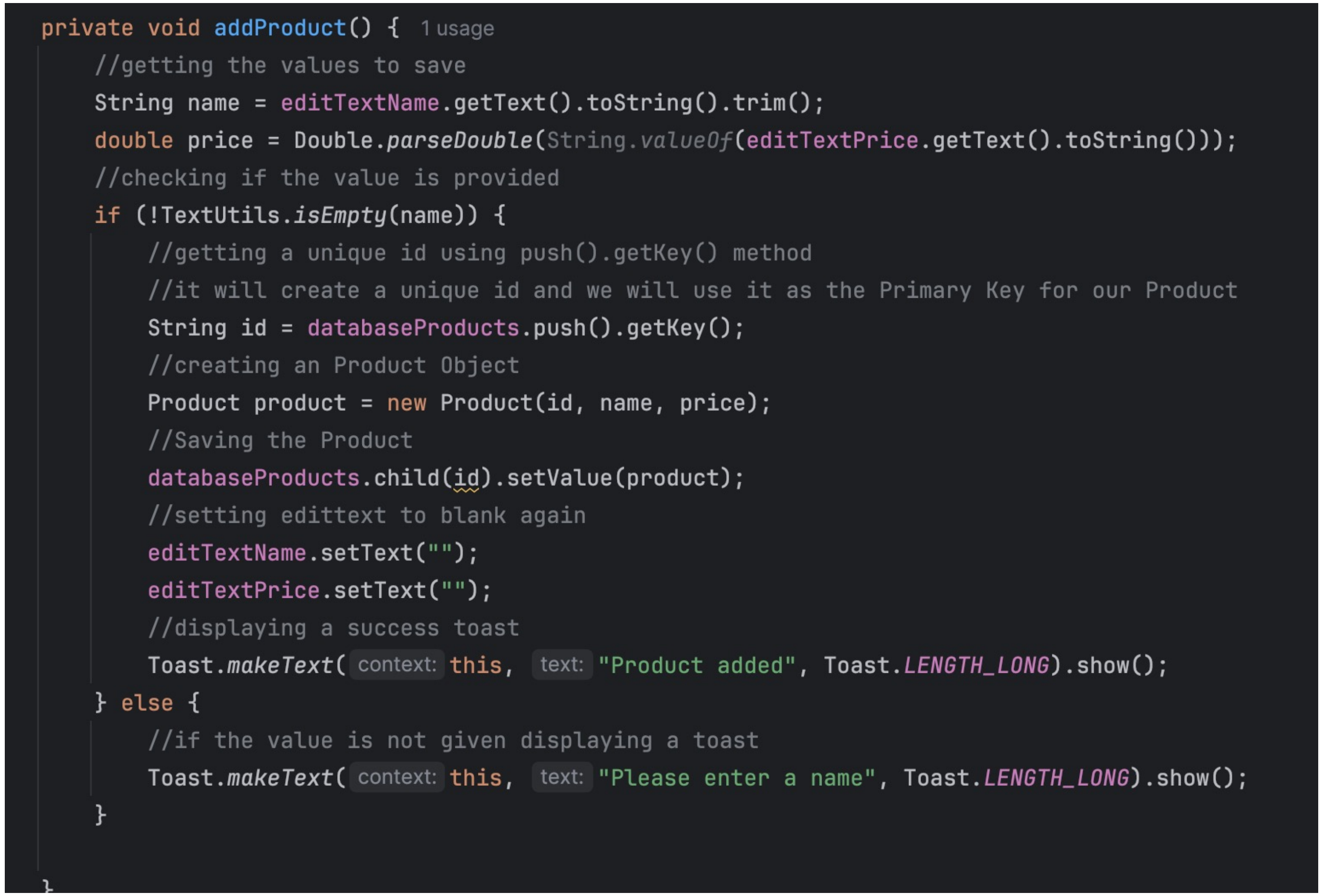
```java
//creating an Product Object
Product product = new Product(id, name, price);
//Saving the Product
databaseProducts.child(id).setValue(product);
```

- Clear the TextBoxes.



```java
//setting edittext to blank again
editTextName.setText("");
editTextPrice.setText("");
```

Add a Product to the Database (final look)



```
private void addProduct() {  1 usage
    //getting the values to save
    String name = editTextName.getText().toString().trim();
    double price = Double.parseDouble(String.valueOf(editTextPrice.getText().toString()));
    //checking if the value is provided
    if (!TextUtils.isEmpty(name)) {
        //getting a unique id using push().getKey() method
        //it will create a unique id and we will use it as the Primary Key for our Product
        String id = databaseProducts.push().getKey();
        //creating an Product Object
        Product product = new Product(id, name, price);
        //Saving the Product
        databaseProducts.child(id).setValue(product);
        //setting edittext to blank again
        editTextName.setText("");
        editTextPrice.setText("");
        //displaying a success toast
        Toast.makeText( context: this,  text: "Product added", Toast.LENGTH_LONG).show();
    } else {
        //if the value is not given displaying a toast
        Toast.makeText( context: this,  text: "Please enter a name", Toast.LENGTH_LONG).show();
    }

}
```

Update a Product on the Database

- Get the string id, name and the price.

```
private void updateProduct(String id, String name, double price) {  1 usage
```

- As the unique id, get the product reference.

```
//getting the specified product reference
DatabaseReference dR = FirebaseDatabase.getInstance().getReference( path: "products").child(id);
```

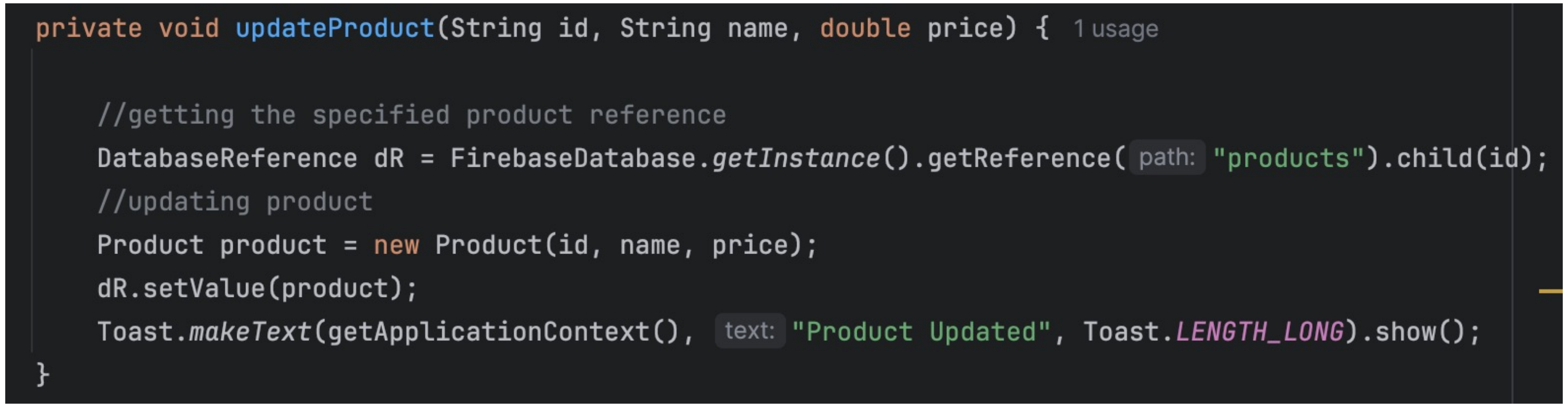
- Update the product by using setValue().

```
//updating product
Product product = new Product(id, name, price);
dR.setValue(product);
```

- Show a success message.

```
Toast.makeText(getApplicationContext(),  text: "Product Updated", Toast.LENGTH_LONG).show();
```

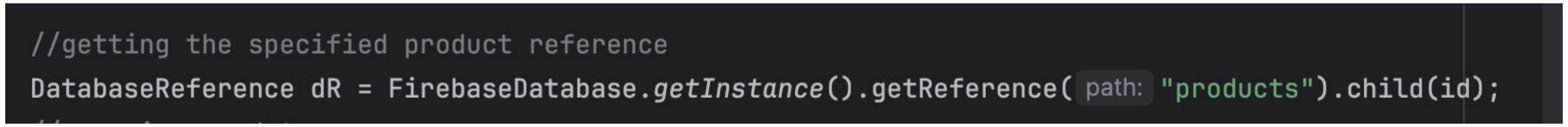
Update a Product on the Database (final look)



```
private void updateProduct(String id, String name, double price) {  1 usage

    //getting the specified product reference
    DatabaseReference dR = FirebaseDatabase.getInstance().getReference( path: "products").child(id);
    //updating product
    Product product = new Product(id, name, price);
    dR.setValue(product);
    Toast.makeText(getApplicationContext(),  text: "Product Updated", Toast.LENGTH_LONG).show();
}
```

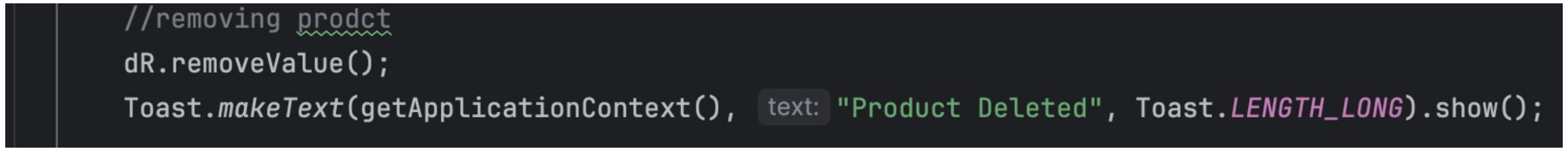
Delete a Product from the Database

- As the unique id, get the product reference.



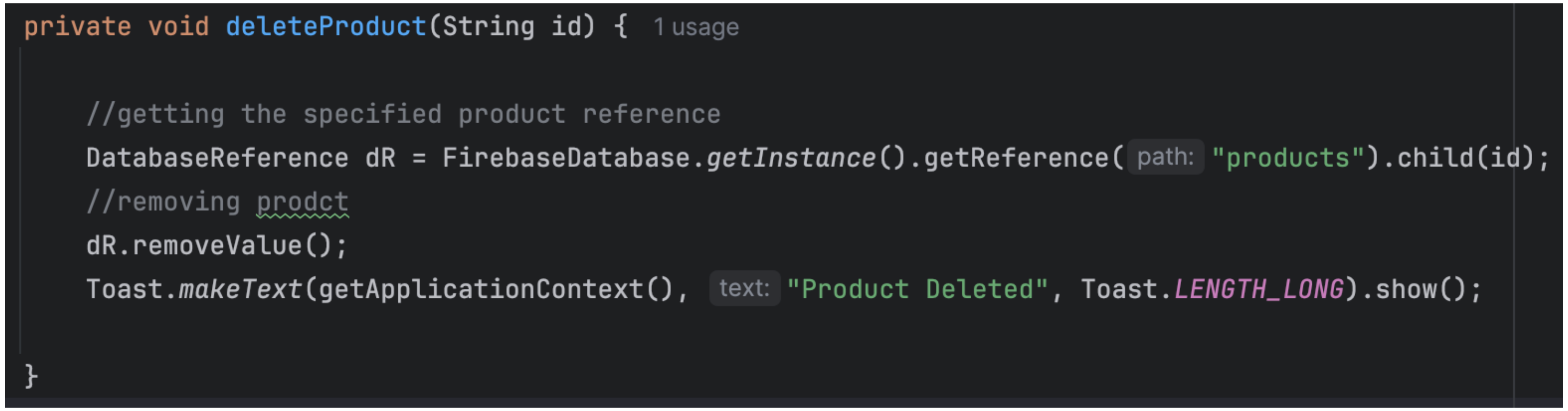
```
//getting the specified product reference
DatabaseReference dR = FirebaseDatabase.getInstance().getReference( path: "products").child(id);
```

- Remove the product and show a success message.



```
//removing prodct
dR.removeValue();
Toast.makeText(getApplicationContext(),  text: "Product Deleted", Toast.LENGTH_LONG).show();
```

Delete a Product from the Database (final look)



```
private void deleteProduct(String id) {  1 usage

    //getting the specified product reference
    DatabaseReference dR = FirebaseDatabase.getInstance().getReference( path: "products").child(id);
    //removing prodct
    dR.removeValue();
    Toast.makeText(getApplicationContext(),  text: "Product Deleted", Toast.LENGTH_LONG).show();

}
```

THANK YOU!