**C302 Web Services: P10 Worksheet**

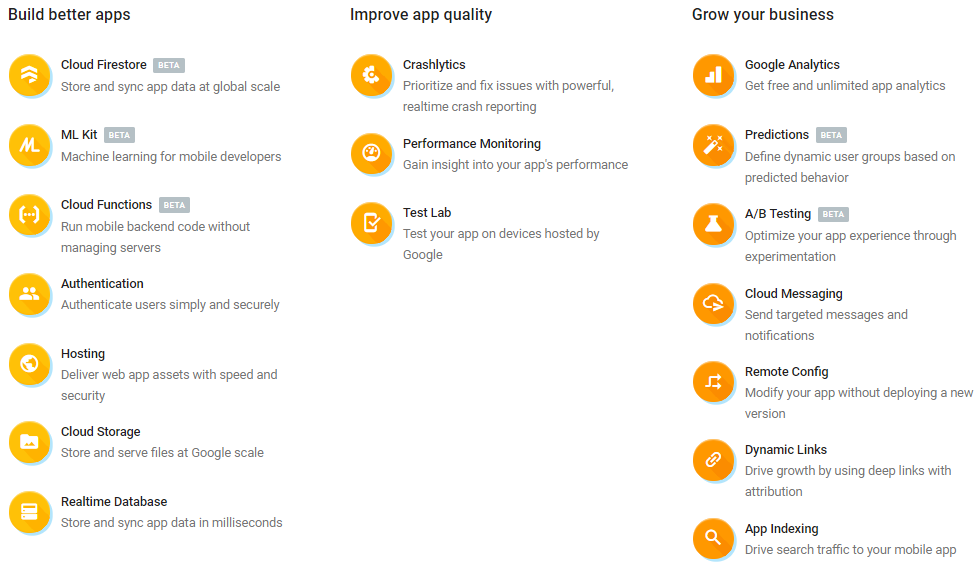
Firebase Cloud





# Introduction to Firebase

Firebase is a powerful platform for building mobile and web app, offering realtime data storage & synchronization, user authentication and more.



Check out Google Firebase at the official website <https://firebase.google.com/>.

# Setting up Android Project

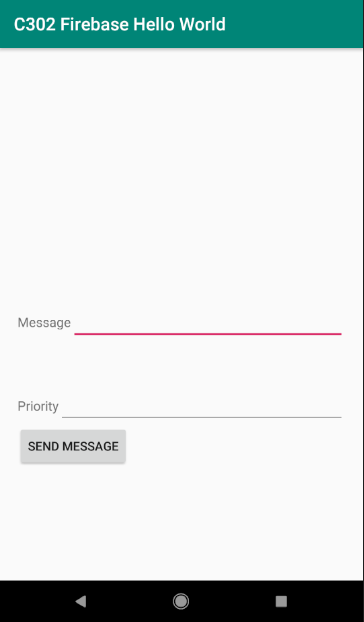
In this section, we will create a “Hello World” Firebase Android app using the source codes provided in your problem package. You can choose to watch “**Set up Android Project**” video in LEO or perform the steps in this section of the worksheet.

1. Launch Android Studio and start a new project

This table below summarizes the info to be entered for the project:

|  |  |  |  |
| --- | --- | --- | --- |
| **Properties** | **Description** | | |
| Application Name | C302 Firebase Hello World | | |
| Company Domain | webservices.rp.edu.sg | | |
| Package Name | sg.edu.rp.webservices.c302firebasehelloworld *(default value will do)* | | |
| Project Location | D:\C302\Workspace\C302FirebaseHelloWorld *(or somewhere suitable)* | | |
| Target Android Device & Min Req SDK | Phone and Tablet / API 16 | | |
| Activity to be added | Empty Activity | Activity Name | MainActivity |
| Layout Name | activity\_main |

1. Go to this package’s “source\Android\java” and use the \*.java files to overwrite those created in your project. Do the same for the \*.xml files in “source\Android\res” to create the layout files.
2. Run the emulator and you should see a screen like this:



# Getting Started with Firebase (Android app)

This section shows you how to add Firebase to your Android project. You can choose to watch “**Add Firebase to Android Project**” and “**Create Firestore database**” videos in LEO or refer to the Firebase Guide.pdf in your student resources. Perform all sections there except “Adding apps to the same project”.

1. What are the configuration files required to be added/updated?

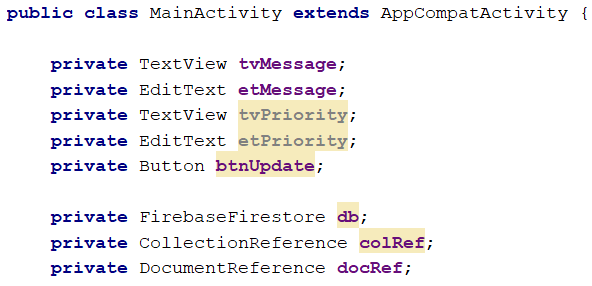
|  |
| --- |
| google.service.json |

# Listening for realtime updates

In this section, we are going to update the Android app so that we can retrieve the message “Hello World” from Firestore. By using Firestore, any change to the message would result in the screen being “refreshed” to reflect the latest message. You can choose to watch “**Listen for realtime updates**” video in LEO or perform the steps in this section of the worksheet

Resource: <https://firebase.google.com/docs/firestore/>

1. Open MainActivity.java and add in the following code to declare the following private variables.



1. Add in the following codes in the onCreate() method to add a snapshot listener which creates a document snapshot with the current contents of the single document.

|  |  |
| --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18** | **db** = FirebaseFirestore.*getInstance*();  **colRef** = **db**.collection(**"messages"**); **docRef** = **colRef**.document(**"message"**); **docRef**.addSnapshotListener(**new** EventListener<DocumentSnapshot>() {  @Override  **public void** onEvent(@Nullable DocumentSnapshot snapshot,  @Nullable FirebaseFirestoreException e) {  **if** (e != **null**) {  **return**;  }   **if** (snapshot != **null** && snapshot.exists()) {  String text = (String) snapshot.get(**"text"**);  **tvMessage**.setText(text);  }  } });  This is your Firebase database name. So change it according to what you named it |

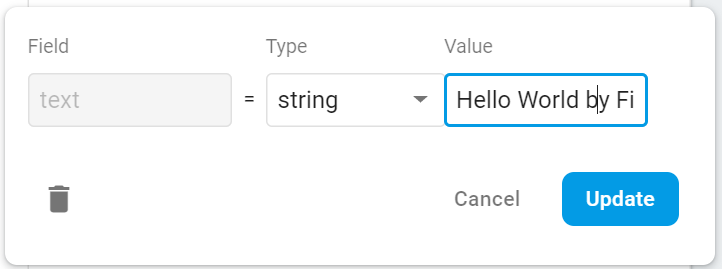
Line 1: Get a FirebaseFirestore instance

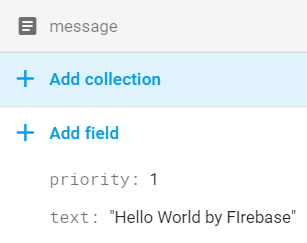
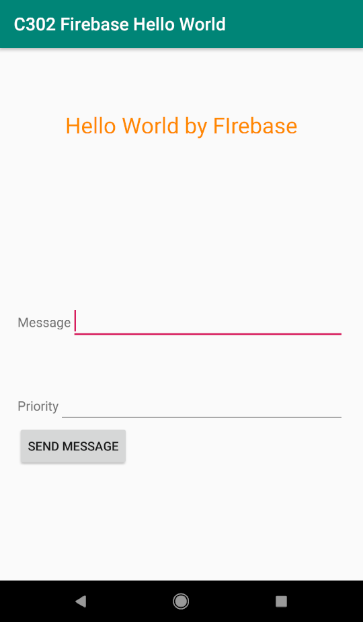
Line 3-4: See explanation below

Line 5-18: Add a snapshot listener to listen for realtime updates. See <https://firebase.google.com/docs/firestore/query-data/listen>

Line 14: retrieve the “text” field of document “message” using getData() or other get() methods in DocumentSnapshot class. See <https://firebase.google.com/docs/reference/android/com/google/firebase/firestore/DocumentSnapshot>

1. Modify the text in Firestore console and press “Update”.



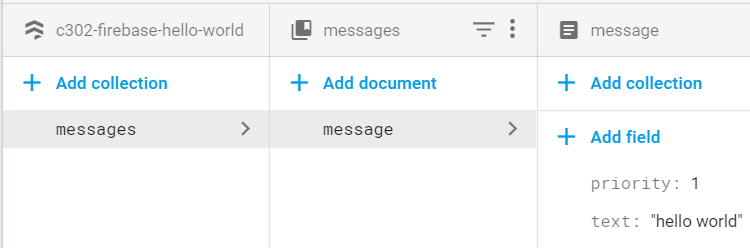
Reflected immediately on Android app

The updated text will be “refreshed” to show the latest text from the Cloud Firestore.

The Android app needs to make a document reference to the Firebase database.

Every document in Cloud Firestore is uniquely identified by its location within the database. The previous example showed a document message within the collection messages. To refer to this location in your code, you can create a reference to it.

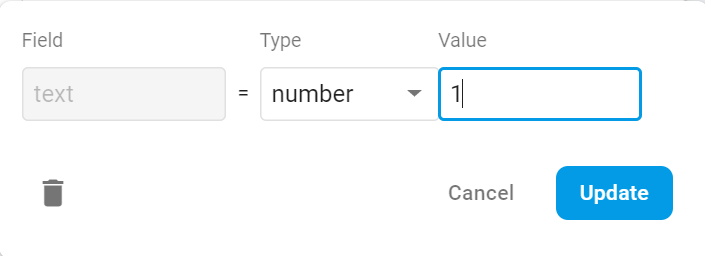




Resources: <https://firebase.google.com/docs/firestore/data-model>

<https://firebase.google.com/docs/firestore/manage-data/structure-data>

1. Change the value of “text” to a number. What is the output from your emulator? Explain your observation.



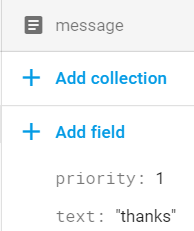
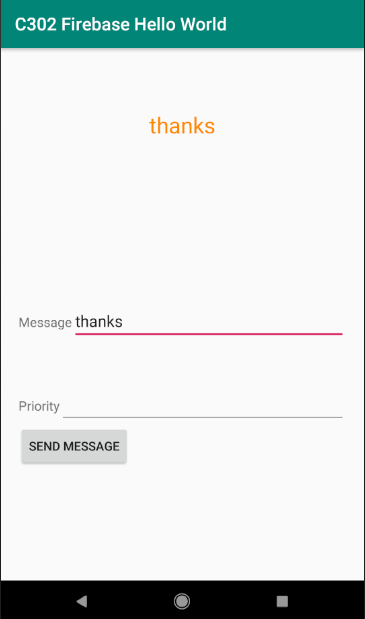
|  |
| --- |
| Change to 1 |

1. What are the kind of values that Firestore can store? Explore the different data types that you can use in Firebase.

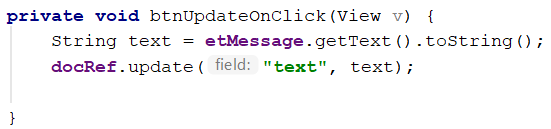
|  |
| --- |
| String, number, Boolean, array, map, null, timestamp, reference |

# More with Firestore (Android app)

In this section, we are going to update the “Hello World” app so that it is able to update the message via user interface (UI) using EditText.

1. In order to update the message in Firebase database, add the following codes into btnUpdateOnClick() method.



Resource: <https://firebase.google.com/docs/firestore/manage-data/add-data#update-data>

docRef.update(**"text"**, text) uses the reference of the document “message” to update the field “text” with the new value.

1. Try changing the following line of code from

docRef.update(**"text"**, text)

to

docRef.update(**"text"**, text, **"color"**, **"red"**)

Explain your observation in the database.

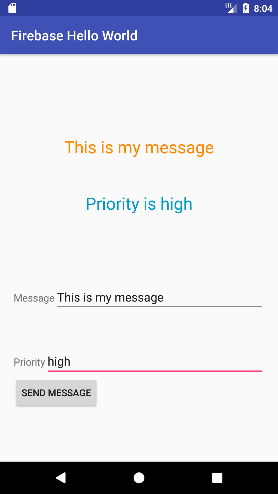
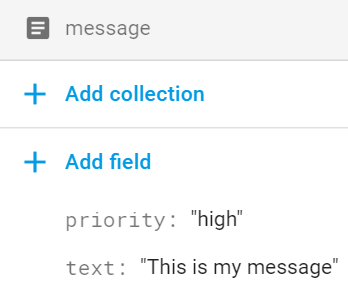
|  |
| --- |
| New field color with value red inside |

# More with Firebase database and POJO

**POJO** stands for Plain Old Java Object. A POJO class has properties, getters and setters for its properties. In Firebase, it is used for data modelling.

|  |
| --- |
| This class must have a **default constructor** that takes no argument and has public getters for the properties to be assigned. |

In the “Hello World” app, it is possible to store more than just the message text. This can be done easily by using POJO.

1. Open the POJO class Message.java. It should have “priority” and “text” properties. Notice that there is a default constructor that does not take in any arguments. Such a constructor is needed for calls to snapshot.toObject(Message.class)
2. Change the code in the addSnapshotListener callback function in MainActivity.java.

From

String text = (String) snapshot.get(**"text"**);

**tvMessage**.setText(text);

To

Message msg = snapshot.toObject(Message.**class**);*//use POJO***tvMessage**.setText(msg.getText());  
**tvPriority**.setText(**"Priority is "**+msg.getPriority());

1. Change the code in the btnUpdateOnClick() method in MainActivity.java.

From

String text = **etMessage**.getText().toString();

docRef.update(**"text"**, text, **"priority"**, **"red"**)

To

String text = **etMessage**.getText().toString();  
String priority = **etPriority**.getText().toString();  
Message msg = **new** Message(text, priority);

docRef.set(msg);

Resource: <https://firebase.google.com/docs/reference/android/com/google/firebase/firestore/DocumentSnapshot>

**End of worksheet**