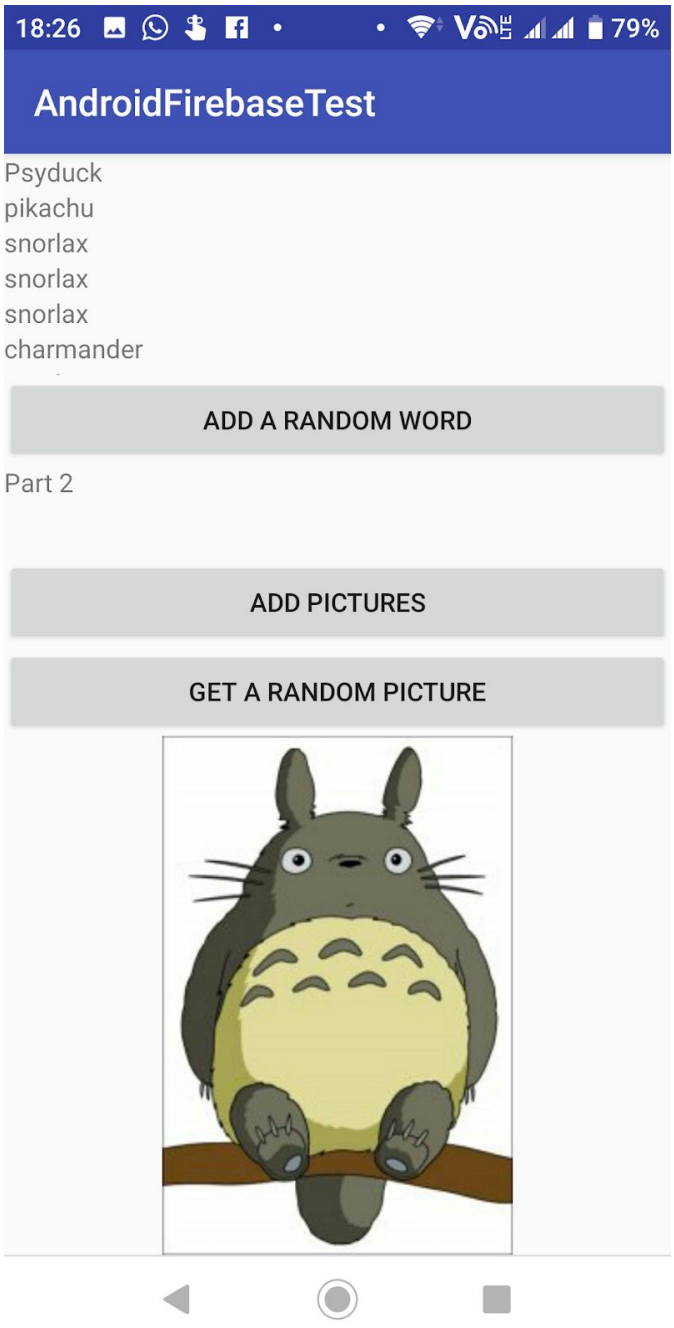
Lesson 5 - Firebase

Introduction

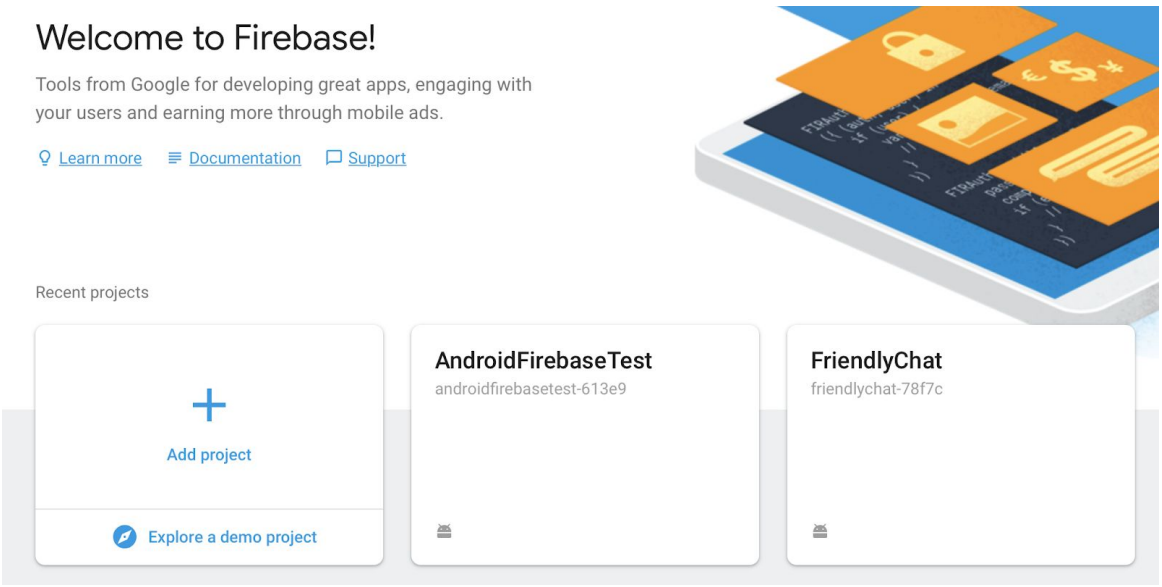
This is a simple app that introduces the code that you need to write to read/write data to firebase.



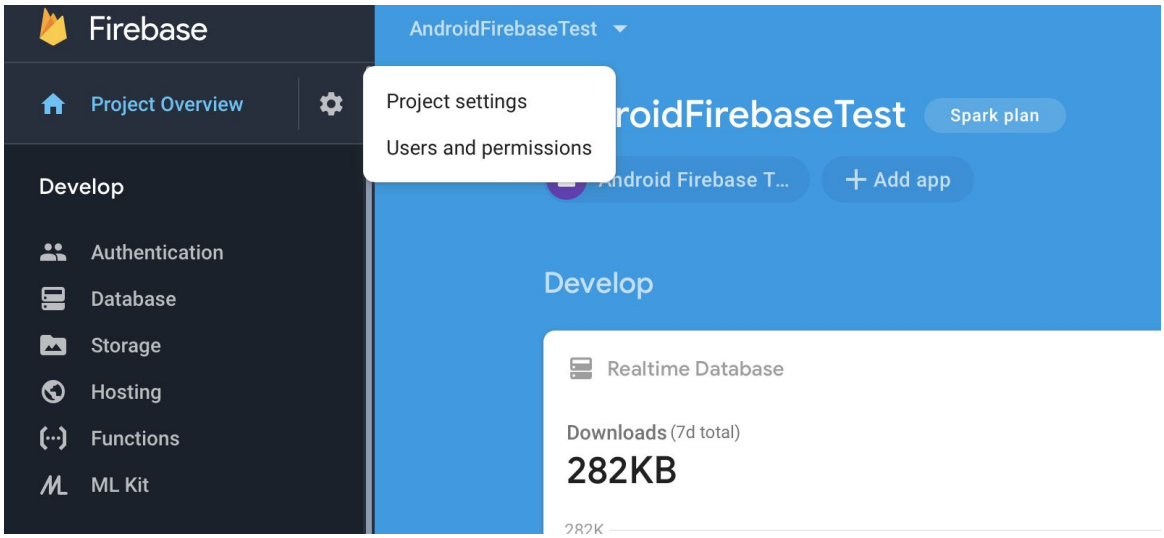
Set up your firebase project

1. Start a new project in Android studio, you will need it in Step 5. Launch a Starter project if you are provided with one.

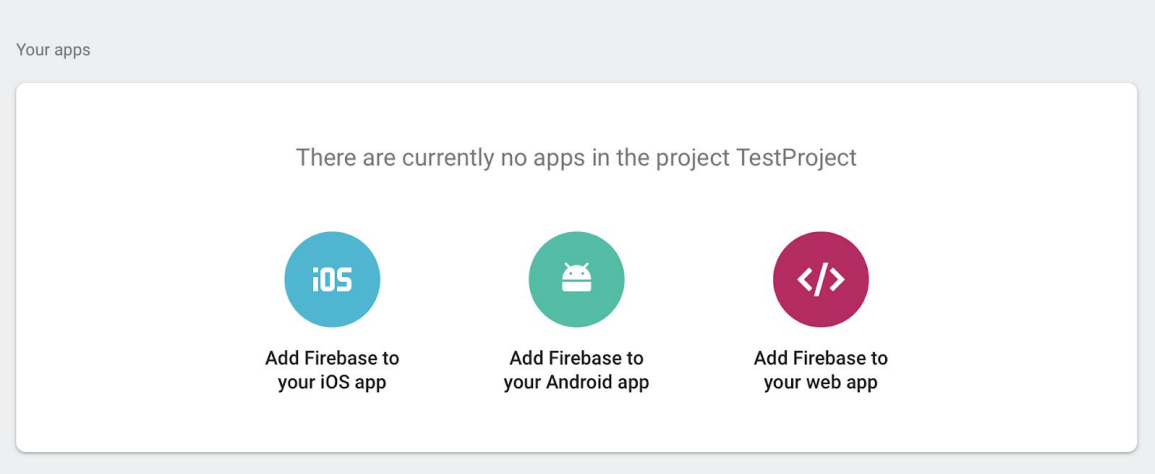
2. Go to console.firebase.google.com and create a new project by selecting Add A project.



3. Find **Project Overview** and click on the icon beside it, then select **Project Settings**.



4. Select the **General Tab**, and scroll down and look for **Add Firebase To Your Android App**.



5. You are required to enter some information:
    a. The package name. Go to your android studio project and get the package name.
    As an example, my package name is **`com.example.norman_lee.androidfirebasetest.`** Use your own.
    b. SHA1 fingerprint: please see instructions here https://stackoverflow.com/questions/15727912/sha-1-fingerprint-of-keystore-certificate

6. The website is user friendly and will tell you what to do next. In general,
    a. You are given a **google-services.json** file to be put in your project. Be sure to put it in the correct location.
    b. You are also told how to modify the gradle files. Please note that the **`compile`** keyword has been replaced by **`implementation`**.
    c. In Android Studio, run your project on your emulator or a phone. If you have followed all the instructions, the website will congratulate you.

7. You are now ready to add firebase's Realtime Database to your project. Click on **Database** on the left-hand menu, and from the database options on the right, look for **Realtime Database**. Select launch in **Test Mode**.
   Test mode removes the need for authentication and simplifies our programming task. Of course, it is not secure and anyone with the same information can connect to your database.

8. Add the following line to the **`dependencies`** section of the **app-level build.gradle** file. (Choose Android view, this is the build.gradle file under the app folder)

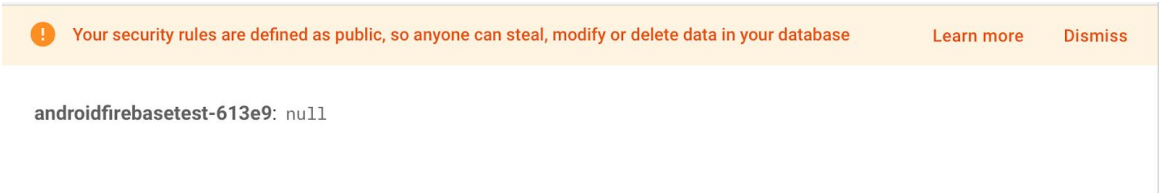
```
implementation 'com.google.firebase:firebase-database:16.0.4'
implementation 'com.google.firebase:firebase-storage:16.0.4'
```

Initialize your databases

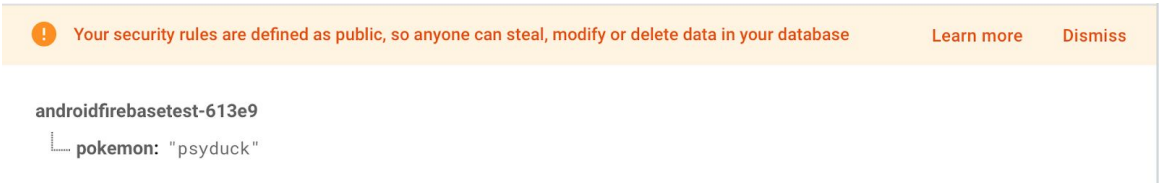
Set up the Firebase Realtime Database.

This is the **Database** item on the left-hand menu.

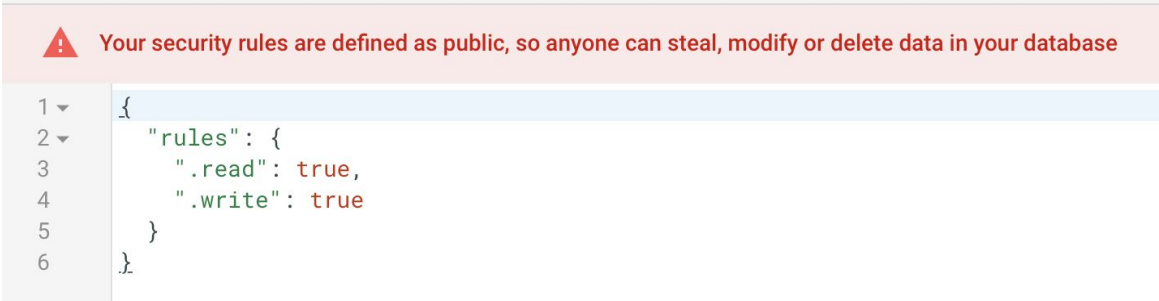
Initially your database will be empty.



Set it up so that it will contain one key-value pair.



Click on the **Rules** tab and check that you see the following. This eliminates the need to provide any login on your app.



Set Up The Storage Database

This is the **Storage** item on the left-hand menu and is meant for data such as images.
Follow the instructions on the screen.

Click on the **Rules** tab and ensure the code within looks like this.
Again, this is to remove the need for authentication.

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      //allow read, write: if request.auth != null;
      allow read, write;
    }
  }
}

```

Part 1 - Pushing and Pulling Text, Dynamic Layout

Setting Up

- Download the starter code, which consists of MainActivity and the layout file.
- Copy and paste this code over the android studio-generated code in the project.
- Put any images you like into the drawables folder.

TODO 10.0 Example

The following code is included to help you check if you have a working connection to firebase without the need to authenticate, and to illustrate to you the structure of the code. You should see “Psyduck” displayed on the top of the screen.
In the firebase database, you should see one key-value pair.

```
textViewSampleNodeValue =
findViewById(R.id.textViewSampleNodeValue);
databaseReferenceSampleNodeValue =
mRootDatabaseRef.child(SAMPLE_NODE);
databaseReferenceSampleNodeValue.setValue("Psyduck");

databaseReferenceSampleNodeValue.addListenerForSingleValueEvent(
        new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {
                textViewSampleNodeValue.setText((String)
dataSnapshot.getValue());
            }

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

            }
        }
);
```

TODO 10.1 Get the reference to the root node of the firebase database

```
DatabaseReference mRootDatabaseRef =
FirebaseDatabase.getInstance().getReference();
```

TODO 10.2 initialize the array of strings

Of course you may choose your own strings:

```
        randomStrings = new ArrayList<>();
        randomStrings.add("pikachu");
        randomStrings.add("snorlax");
        randomStrings.add("charmander");
```

TODO 10.3 Get a reference to the child node

If the child node does not exist, then one is created.

The name of the child node is stored as a string constant.

```
databaseReferencePart1 = mRootDatabaseRef.child(CHILD_NODE_PART1);
```

TODO 10.4 Get a reference to the “Add a Random Word” button, set up the OnClickListener and upload a random word to firebase.

In the code below, we upload values to firebase, and let firebase generate the key for us automatically. The key will be a string of characters.

```
buttonPart1 = findViewById(R.id.buttonPart1);

buttonPart1.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {

       //Select a random string
       Random random = new Random();
       int position = random.nextInt(randomStrings.size());

databaseReferencePart1.push().setValue(randomStrings.get(position)
);

   }
});
```

TODO 10.5 In onStart(), we listen for changes in the database and dynamically update the scrollView widget with textView widgets

Listening to changes in the database is done by passing an anonymous instance of **`ValueEventListener()`** to the **`addValueEventListener`** method.

You specify what happens in **`onCancelled`** if the download is not successful.

Within **`onDataChange`**

- A reference to the LinearLayout widget (within the ScrollView widget) is obtained.
- All child views are removed
- All the child key-value pairs within the **`Part1`** node in the firebase database are passed to the **`dataSnapshot`** parameter
- We loop through each of the children to get the respective values.
- Each value is placed as a new TextView widget

```
@Override
protected void onStart() {
   super.onStart();
   databaseReferencePart1.addValueEventListener(new
ValueEventListener() {
       @Override
       public void onDataChange(@NonNull DataSnapshot
dataSnapshot) {
           LinearLayout linearLayout =
findViewById(R.id.linearLayoutPart1);
           linearLayout.removeAllViews();
           for(DataSnapshot ds : dataSnapshot.getChildren()){
               Log.i(TAG, ds.getKey());
               Log.i(TAG, (String) ds.getValue()) ;
               TextView textView = new
TextView(MainActivity.this);
               textView.setText((String) ds.getValue());
               linearLayout.addView(textView);
           }
       }
       @Override
       public void onCancelled(@NonNull DatabaseError
databaseError) { }
   });
```

Part 2 - Pushing and Pulling Images

Strategy

Images will be uploaded to Firebase Storage under the images folder.

Information on the images will be stored in Firebase database as follows:

TODO 10.6 Get a reference to the root note of the firebase storage

```
StorageReference storageReference =
FirebaseStorage.getInstance().getReference();
```

TODO 10.7 Write a static inner class for Part2

This will act as the key-value pairs at the lowest level of the json tree.

```
    static class ImageData{

        String filename;
        String description;

        ImageData(String filename, String description){
            this.filename = filename;
            this.description = description;

        }
    }
```

TODO 10.8 Build a HashMap object with your data

```
final Map<String, ImageData> imageDataMap = new HashMap<>();
imageDataMap.put("Miku", new ImageData("hatsunemiku",
"vocaloid"));
imageDataMap.put("Pikachu", new ImageData("pikachu", "pokemon"));
imageDataMap.put("Totoro",new ImageData("totoro", "studio
ghibli"));
```

TODO 10.9 Get reference to the root of the child node part 2

```
databaseReferencePart2 = mRootDatabaseRef.child(CHILD_NODE_PART2);
```

TODO 10.10 Get reference to the Add Pictures button and write code to upload the HashMap data when button is clicked

Uploading your hashMap just requires one line.

```
buttonPart2AddPicture = findViewById(R.id.buttonPart2AddPicture);

buttonPart2AddPicture.setOnClickListener(new
View.OnClickListener() {
   @Override
   public void onClick(View view) {

       databaseReferencePart2.setValue(imageDataMap);
       // Complete TODO 10.11 here

   }
});
```

TODO 10.11 Loop through each entry in the hashmap and do the necessary to upload the image to firebase

For each entry in the HashMap, I get the filename and build the filepath for storage into firebase.
These two information is passed to the **`uploadFileToFirebaseStorage`** method.
I then manually add a new key-value pair to store the filepath.

```
       for( String key: imageDataMap.keySet()){

           ImageData imageData = imageDataMap.get(key);
           String path = "images/" + imageData.filename + ".jpg";
           uploadFileToFirebaseStorage(imageData.filename,path);

databaseReferencePart2.child(key).child("path").setValue(path);

       }
```

TODO 10.12 Get a reference to the widgets and write code to download an image randomly when the Get Picture button is clicked

- First, get the keys from the HashMap and select a random key.
- Next we invoke the addListenerForSingleValueEvent and write an anonymous instance of ValueEventListener to download the image

```
buttonPart2GetPicture = findViewById(R.id.buttonPart2GetPicture);
imageViewPart2 = findViewById(R.id.imageViewPart2);

buttonPart2GetPicture.setOnClickListener(new
View.OnClickListener() {
   @Override
   public void onClick(View view) {

          //Get a Random Key
          //Invoke addListenerForSingleValueEvent to download the
image

   }

}
);
```

Get a Random Key

```
       ArrayList<String> keys = new ArrayList<>(imageDataMap.keySet());
       Random r = new Random();
       int position = r.nextInt(keys.size());
       final String searchKey = keys.get(position);
       Log.i(TAG, keys.get(position));
```

Downloading the image

```
databaseReferencePart2.addListenerForSingleValueEvent(new
ValueEventListener() {

    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot){

         for (DataSnapshot ds: dataSnapshot.getChildren()){
             Log.i(TAG, "key:" + ds.getKey());
             if( searchKey.equals( ds.getKey() ) ){
                  Log.i(TAG, "path:"+ ds.child("path").getValue() );
                  downloadFromFirebaseStorage(
                              (String) ds.child("path").getValue(),
                              imageViewPart2);
                   }
          }

    }

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

}
```