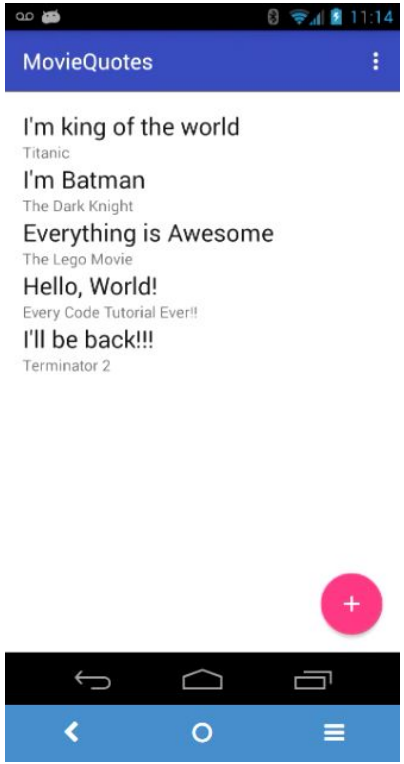
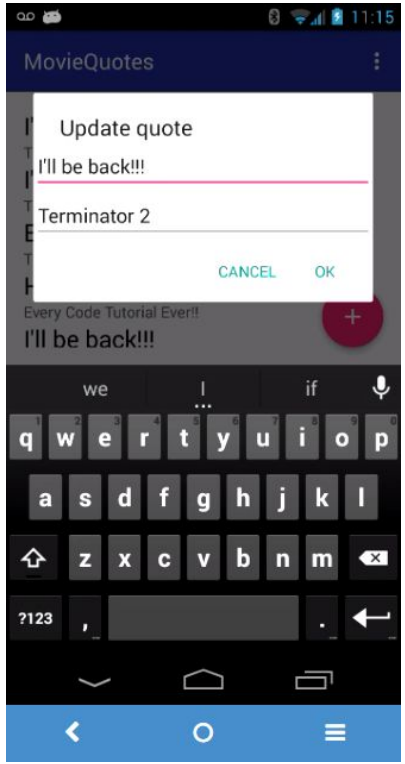
# Introduction to using Firebase v3 in Android


```
boutell-movie-quotes
  quotes
    -JnHqRle6xTHMC2crgoL
      movie: "Terminator 2"
      quote: "I'll be back!"
    -JnHrxl7U59xhKKKOuXk
      movie: "Every Code Tutorial Ever..."
      quote: "Hello, World!"
    -JnlatYwzVhBmm3GPMk9
      movie: "The Lego Movie"
      quote: "Everything is Awesome"
    -JnIb7oeomj9w3S6XqLz
      movie: "The Dark Knight"
      quote: "I'm Batman"
    -JnM84yjWL1q0xcVb-le
      movie: "Titanic"
      quote: "I'm king of the world"
    -JnOt5RkW0uLsNOatgdE
      movie: "Monty Python and the Holy Grail"
      quote: "She turned me into a newt!"
    -KMRoL9gKasmqa2Sr0CJ
      movie: "Indiana Jones and the Raiders of the Lost Ark
      quote: "The man is nefarious"
```




ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Acknowledgment

A previous version of these slides was created by Tyler Rockwood during an independent study course.  So when you see the username **rockwotj** in places that's why.

https://plus.google.com/+TylerRockwood9/posts

# By the end of this unit you should be able to...

- ❑ Connect an Android client to a Firebase realtime database backend
  - ❑ Create a Firebase object
  - ❑ Implement listeners to receive data
  - ❑ Deserialize JSON data to Java classes
- ❑ Create an Android client that can modify the data
  - ❑ Serialize Java classes to JSON data
  - ❑ Implement CRUD methods in Firebase

MovieQuotes? See
http://xkcd.com/1427/

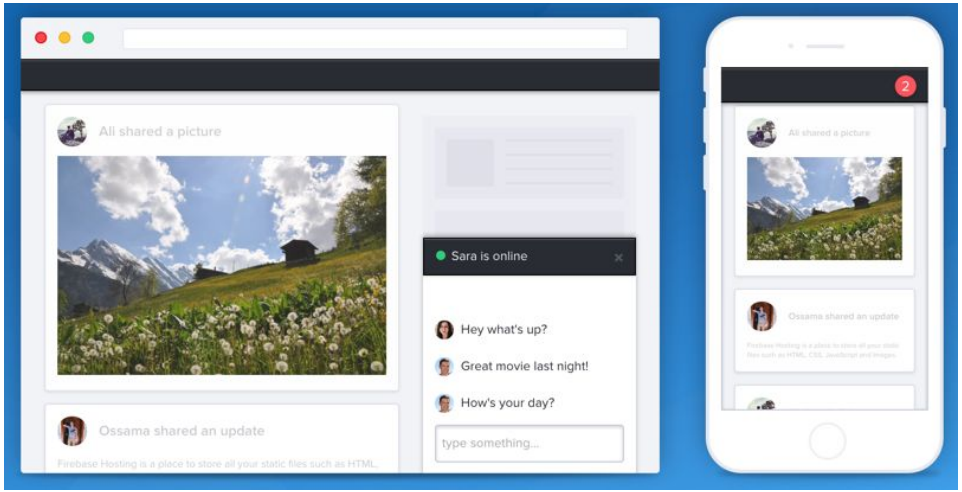
# What is Firebase?

In this lesson you will learn about the history and capabilities of Firebase

# Firebase Realtime database

Firebase (firebase.google.com) is a REALTIME cloud data storage backend that **synchronizes** your data across all devices connected to your Firebase app URL.

- Real-time (sockets)
- Cross-platform
- No-SQL
- Supports authentication
- Has offline capabilities
- Now integrated with storage, cloud messaging, and analytics



# Firebase History

San Francisco based company founded in 2011 by some Rose-Hulman guys and James Tamplin.
The company was acquired by Google in October 2014.
Firebase was rolled out as an integrated Google product at Google I/O 2016.


2006 alumni Michael Lehenbauer and Andrew Lee

http://www.rose-hulman.edu/news/alumni/2015/young-alumni-duo-finds-success-with-firebase-software-development-product.aspx

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Android setup for Firebase

In this lesson we'll download some starting code and add the Firebase framework.

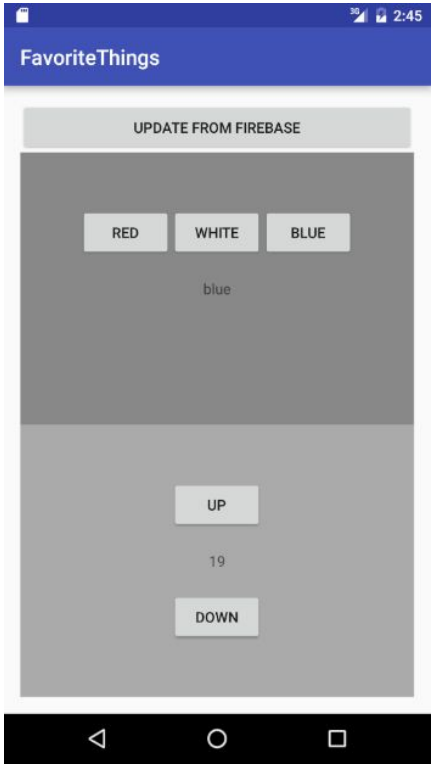

+

# Get the starting code

Download, unzip, and open in Andoid Studio the [starting code](#).

It's a basic MVC app with buttons from week 1.

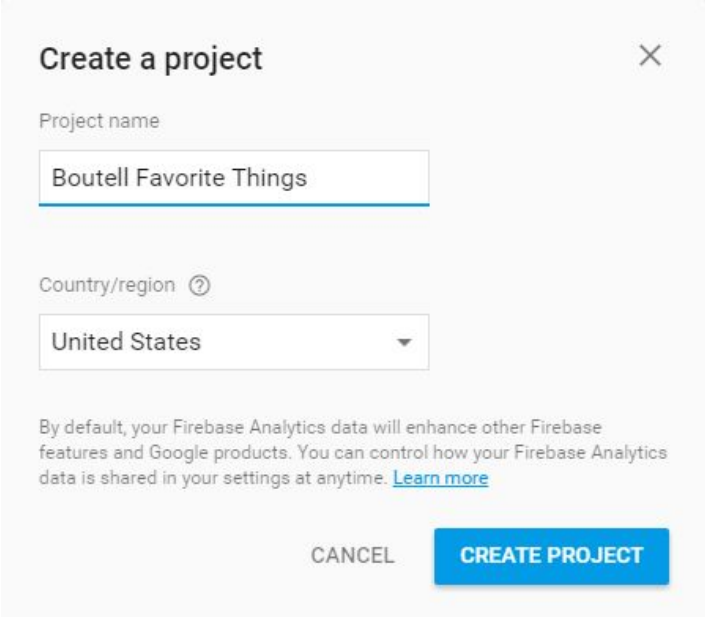
We'll need some files/settings from Firebase, so let's do that now.




ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Firebase setup

In this lesson you will create a Firebase account and prepare a location for saving your favorite things

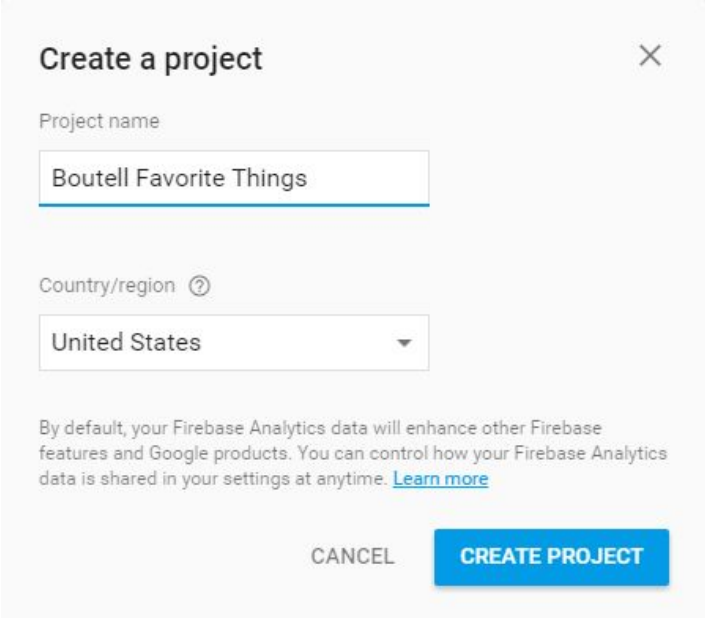


# Getting started on Firebase

Start out by creating an account

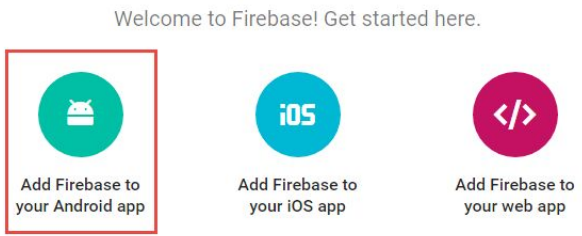
Then go ahead and create a new app with the name '<Your Username> Favorite Things'. It will make a lowercase, spinal-case url like:

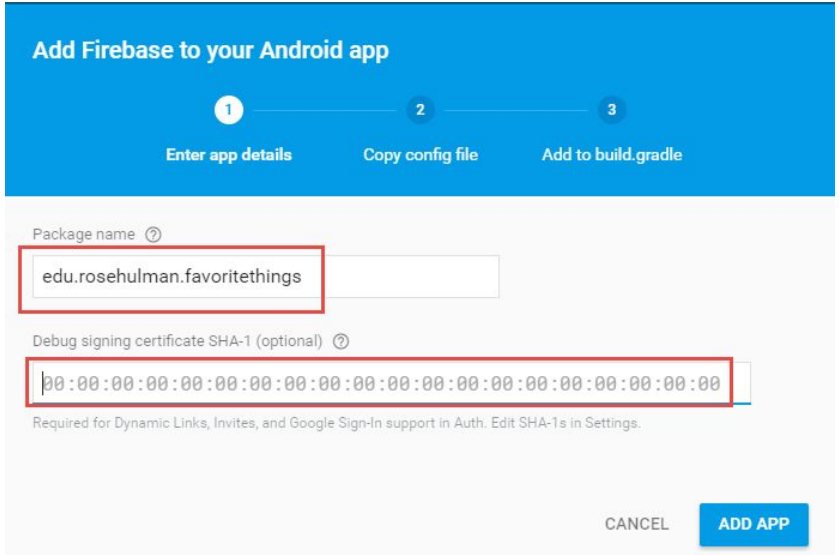
<your username>-favorite-things



# Add Firebase to an existing Android App

When you first create your project or when you click “Add app”, you’ll get this screen → . Click the Android option and follow the instructions.




1. Your Android app’s package.
2. The SHA-1 fingerprint is for Google auth - you can get it from Android Studio (see here).
3. Download google-services.json (details) and copy it into your app
4. Modify your build.gradle, including dependency for **firebase-database**.

Use 10.0.1 in Jan, ‘17 (latest)

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Prepare your Firebase backend: Realtime Database


Use the plus icon to add some simple data directly to your database

Our goal will be to get these values to our app.

# Click the database rules tab and change read/write to true

Firebase

Boutell FavoriteThings2

Boutell FavoriteThing...

Analytics

DEVELOP

Auth

Database

Storage

Hosting

Remote Config

Test Lab

Realtime Database

DATA RULES

Unpublished changes PUBLISH DISCARD SIMULATOR

```
1 {
2   "rules": {
3     ".read": "auth != null",
4     ".write": "auth != null"
5   }
6 }
```

Then publish changes

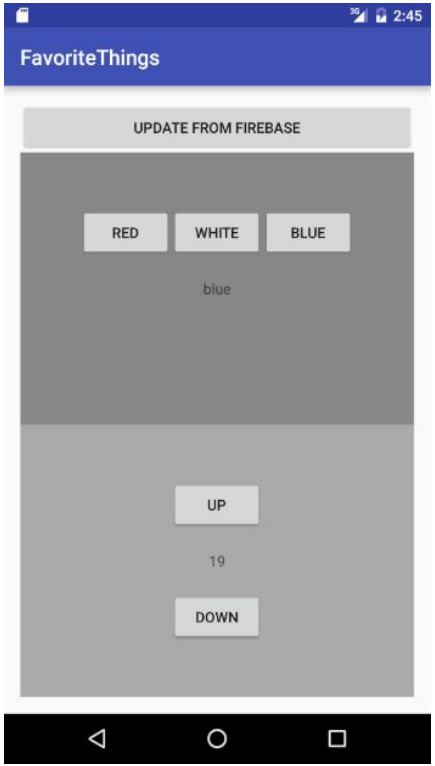
Unpublished changes PUBLISH DISCARD

```
1 {
2   "rules": {
3     ".read": "true",
4     ".write": "true"
5   }
6 }
```

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Pushing data to Firebase and back

In this lesson we'll finish set up and learn how to push data to Firebase and to listen for data added to Firebase.




ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Add Firebase to gradle, internet permission to the manifest

```
<uses-permission android:name="android.permission.INTERNET" />
```

## Add to your app gradle:

```
dependencies {
  ...
  implementation 'com.google.firebase:firebase-database:12.0.1'
}
apply plugin: 'com.google.gms.google-services'
```

## Add to your project gradle:

```
dependencies {
  classpath 'com.google.gms:google-services:3.2.0'
all projects/repositories {
  jcenter()
  google()
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.rosehulman.moviequotes" >

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

**Better**: let the assistant at Tools > Firebase > Realtime Database help you with setup.

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Create a Firebase Reference in MainActivity

```
private long mNumber;
private DatabaseReference mFirebase;
private static final String TAG = "FAVES";

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

    mFirebase = FirebaseDatabase.getInstance().getReference();

    mColorTextView = (TextView) findViewById(R.id.color_text_view);
```

# Set the color to a fixed value

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

    mFirebase = FirebaseDatabase.getInstance().getReference();
    mFirebase.child("color").setValue("Aqua");
```

Then change your buttons so they set the values to red, white, blue **instead of** setting the color in your app.

Do it again for the numbers, but using a different path:

**mFirebase.child(“number”).setValue(17);**

# Now listen for changes from Firebase when we click

```
    return;
case R.id.update_color_button:
    Log.d(TAG, "Updating from Firebase");
    mFirebase.child("color").addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(DataSnapshot dataSnapshot) {
            mColorTextView.setText((String)dataSnapshot.getValue());
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {
            Log.d(TAG, "Database error");
        }
    });
```

**addListenerForSingleValueEvent()** listens *once* and then stops, which is good for one-time-only events. We often instead **addValueEventListener()** to make a listener that continually listens for changes until we remove the listener.

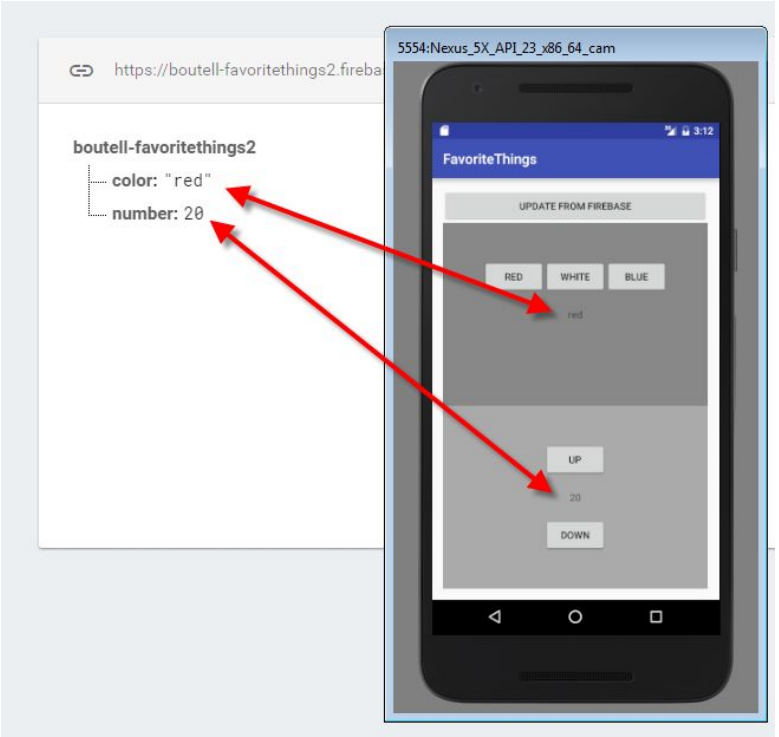
# Do the same for numbers.

This time the snapshot's getValue() is a number, so cast it to a Long to set the field. But remember, TextViews need strings.

```
mNumber = (Long)dataSnapshot.getValue();
mNumberTextView.setText("" + mNumber);
```

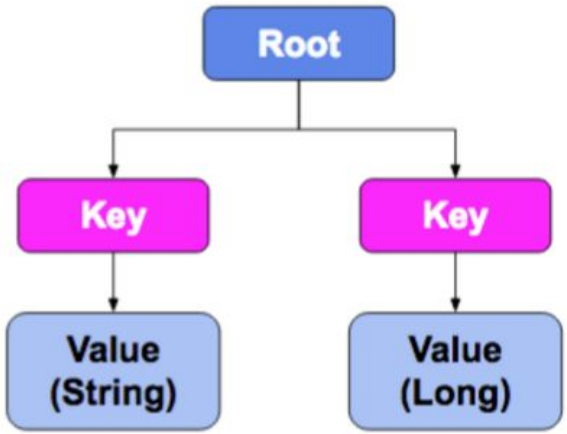
# Run it. Test that changes happen in both directions.

1. Make a change in the app. The backend should update, and the app should too when you press “Update”.
2. Make a change directly on the backend then press update.



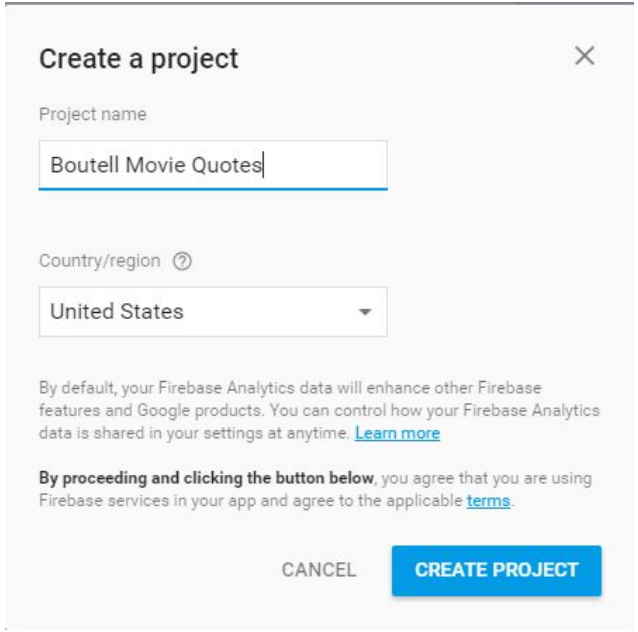
# Firebase data concepts

My colleague Dave Fisher has a nice presentation on how data is stored in Firebase. See his slides. His video is next in Rosebotics (or search "youtube dave fisher firebase concepts").



# Firebase setup for MovieQuotes

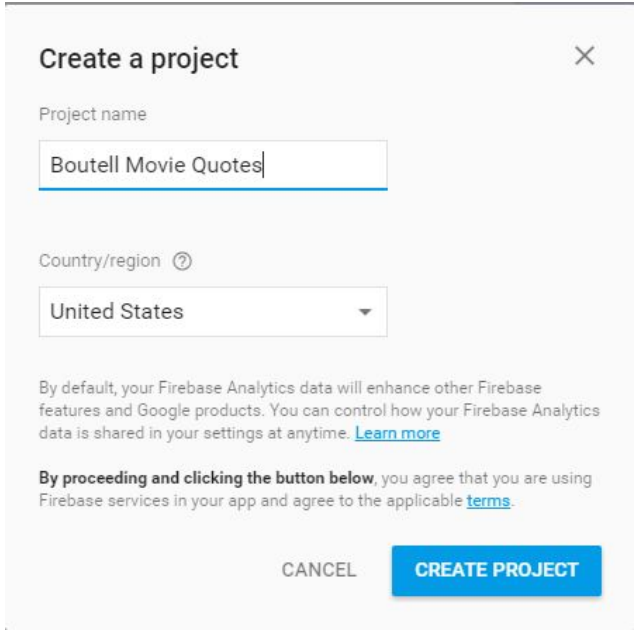
In this lesson you will prepare a location for saving MovieQuotes.



# Getting started on Firebase

Create a new app with the name '<Your Username> Movie Quotes'. It will make a lowercase, spinal-case url like:

<your username>-movie-quotes



# No backend work needed ahead of time

Reminder: inserting to the Firebase will create the table **on the fly**, so you don't need to do any setup of the backend ahead of time!

```
boutell-movie-quotes
  quotes
    -JnHqRle6xTHMC2crgoL
      movie: "Terminator 2"
      quote: "I'll be back!"
    -JnHrxI7U59xhKKKOuXk
      movie: "Every Code Tutorial Ever..."
      quote: "Hello, World!"
    -JnlatYwzVhBmm3GPMk9
      movie: "The Lego Movie"
      quote: "Everything is Awesome"
    -JnIb7oeomj9w3S6XqLz
      movie: "The Dark Knight"
      quote: "I'm Batman"
    -JnM84yjWL1q0xcVb-Ie
      movie: "Titanic"
      quote: "I'm king of the world"
    -JnOt5RkW0uLsNOatgdE + ×
      movie: "Monty Python and the Holy Grail"
      quote: "She turned me into a newt!"
    -KMRoL9gKasmqa2Sr0CJ
      movie: "Indiana Jones and the Raiders of the Lost Ark
      quote: "The man is nefarious"
```

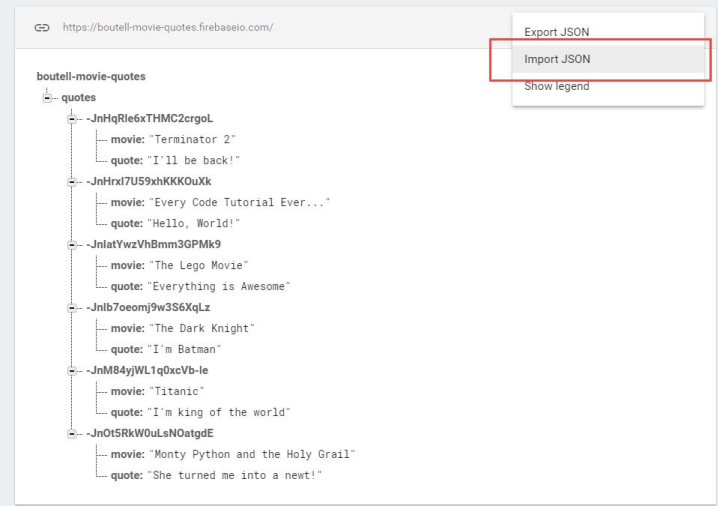
# Adding some hardcoded data (optional)

Import [this data]() into your Firebase app

The import button is in the top right hand corner of the database

After you import the data make a few minor changes to quotes just for practice

You can export in the same way




ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Firebase Security & Rules (copy/paste next slide)

Rules let you specify:

… how your data will be formatted in the NoSQL database.

… who has read and write access to the data. Especially important in a web client!

More info here.

Realtime Database

DATA | RULES

Unpublished changes | PUBLISH | DISCARD

```
{
  "rules": {
    "quotes": {
      ".read": true,
      ".write": true,
      "$moviequote": {
        "movie": { ".validate" : "newData.isString()" },
        "quote": { ".validate" : "newData.isString()" },
        "$other": { ".validate": false }
      }
    }
  }
}
```

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Moviequotes Rules

Copy and paste these rules into your app:

```
{
  "rules": {
    "quotes": {
      ".read": true,
      ".write": true,
      "$moviequote": {
        "movie": { ".validate" : "newData.isString()" },
        "quote": { ".validate" : "newData.isString()" },
        "$other": { ".validate": false }
      }
    }
  }
}
```

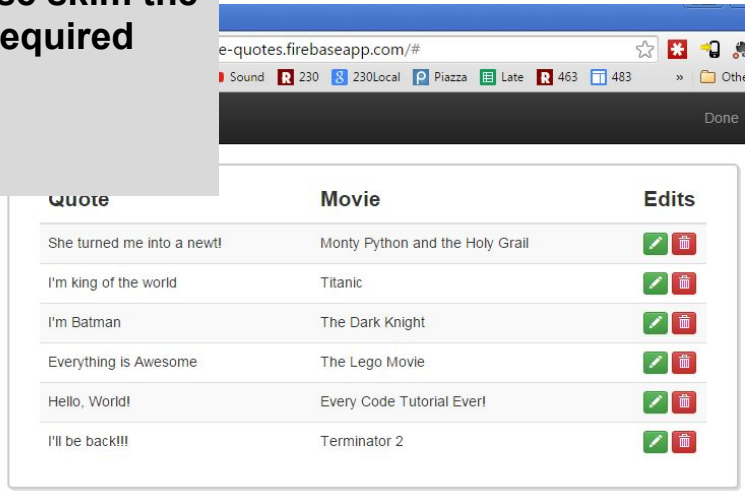
They are porting Firebase Storage Security Rules to the database to simplify this.

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Web Client

In this lesson you will learn how to deploy a finished web a ay to modify the Fireba

**Our web client is broken, so please skim the video, but no other action is required**



# Deploy the web app

In order to deploy the web app you need to download the code from the link below and modify the Firebase app URL.

## Web Client Code

You will need to open **js/app.js** to set **your** Firebase app URL (i.e. **your** username)

## Instructions to deploy the site

Heads up: This will require you to install NodeJS. (which is easy)

Use <**your** username>-movie-quotes for the Firebase app questions

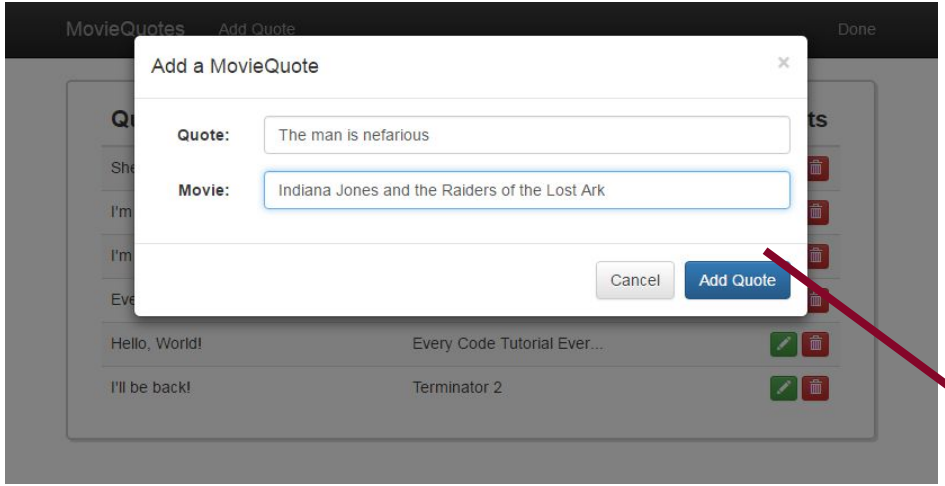
Use the defaults (so don't overwrite the given index.html)

Then fix URL in js/app.js

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Visit your site

Perform some CRUD methods from the web app





```
boutell-movie-quotes
  quotes
    -JnHqRle6xTHMC2crgoL
      movie: "Terminator 2"
      quote: "I'll be back!"
    -JnHrxI7U59xhKKKOuXk
      movie: "Every Code Tutorial Ever..."
      quote: "Hello, World!"
    -JnlatYwzVhBmm3GPMk9
      movie: "The Lego Movie"
      quote: "Everything is Awesome"
    -JnIb7oeomj9w3S6XqLz
      movie: "The Dark Knight"
      quote: "I'm Batman"
    -JnM84yjWL1q0xcVb-Ie
      movie: "Titanic"
      quote: "I'm king of the world"
    -JnOt5RkW0uLsNOatgdE
      movie: "Monty Python and the Holy Grail"
      quote: "She turned me into a newt!"
    -KMRoL9gKasmqa2Sr0CJ
      movie: "Indiana Jones and the Raiders of the Lost Ark
      quote: "The man is nefarious"
```


ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Setup Android Studio for Firebase

In this lesson we'll download some starting code and add the Firebase framework.

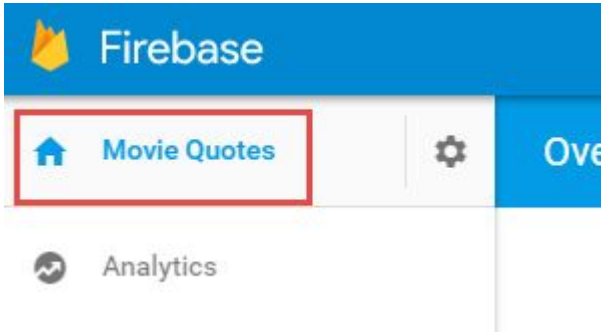

+

# Download the starting code

Download from this repository.

Look it over. Should be similar to code you have written in the past, except the add/edit dialog has a TextWatcher that calls update every time a character is typed.

# Setup Firebase like you did for FavoriteThings




Click on your project in Android Studio

Choose “Tools > Firebase > Realtime Database”.

Then follow the instructions

Don’t forget the gradle dependency (a higher version is OK):

```
compile 'com.google.firebase:firebase-database:10.0.1'
```

Then set rules: read: “true”, write: “true”.

Google is working on a new rules language for security, so don’t worry about rules.

# Add the internet permission to the manifest

```
<uses-permission android:name="android.permission.INTERNET" />
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.rosehulman.moviequotes" >

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

# Firebase needs a key property


```java
public class MovieQuote {
    private String quote;
    private String movie;
    private String key;

    public MovieQuote(String quote, String movie) {
        this.movie = movie;
        this.quote = quote;
    }

    public String getKey() {
        return key;
    }

    public void setKey(String key) {
        this.key = key;
    }

    public String getQuote() { return quote; }

    public void setQuote(String quote) { this.quote = quote; }

    public String getMovie() { return movie; }

    public void setMovie(String movie) { this.movie = movie; }
}
```

Recall, alt-insert to add getter/setter

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Create a Firebase Reference

**Create a database reference** in the constructor of the MovieQuoteAdapter.

```
public class MovieQuoteAdapter extends RecyclerView.Adapter<MovieQuoteAdapter.ViewHolder> {

    private List<MovieQuote> mMovieQuotes;
    private Callback mCallback;
    private DatabaseReference mMovieQuotesRef;

    public MovieQuoteAdapter(Callback callback) {
        mCallback = callback;
        mMovieQuotes = new ArrayList<>();
        mMovieQuotesRef = FirebaseDatabase.getInstance().getReference().child("quotes");

    }
```

# How do we interact with our data?
# We'll implement "CRUD" in next lessons

"CRUD" methods

**C**reate (push)

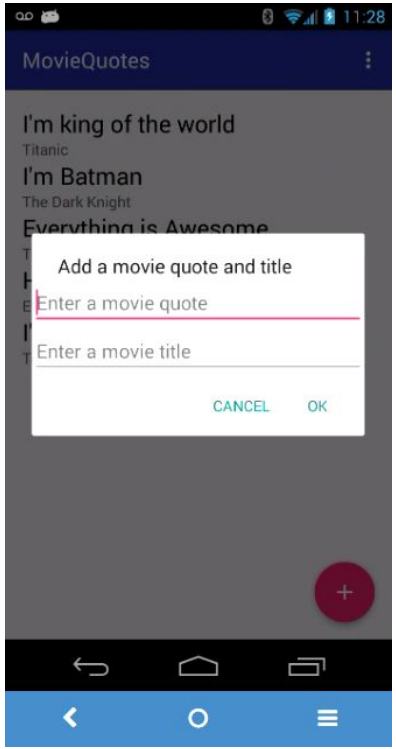
**R**ead (Listener)

**U**pdate (setValue)

**D**elete (removeValue)

# Pushing data to Firebase

In this lesson we'll learn more about the Firebase data format and how to push data to Firebase.




ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Firebase data model

Everything in Firebase is JSON.

JSON is a format to store/send **objects**.

Uses a map to represent objects: fieldname: value

Stored in a tree, accessible by URL path

Can access at any level.

Uses the GSON library to serialize

JSON ↔ Java model object

*if* our model object is json-compliant

Firebase has this built-in, so you just need to know how to use it

GSON is like the Jackson library we've used in past

# Making our model object json-compliant (1)

We want each **model object** to have a key
(Firebase's remove() method needs a key, for example.)

But in Firebase, the key is **separate** from the object

So we'll tell Firebase to ignore the key **on the getter***:

```
public class MovieQuote {
  @Exclude
  public String getKey() {
     return key;
  }
```

and we'll manage it ourselves.

* If your field is public, you exclude the field directly.


ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Making our model object json-compliant (2)

Gson also requires us to have a default (empty) constructor:

```
public MovieQuote() {
    // empty constructor required for Firebase deserialization from JSON
}
```

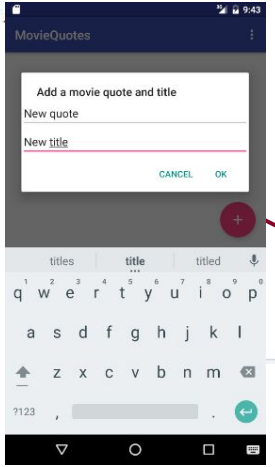
Note: Gson also lets you rename fields:
@SerializedName("movie")
private mMovie;

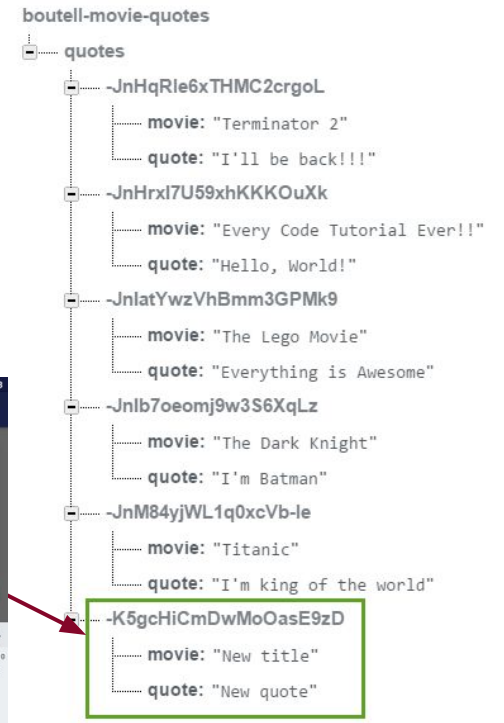
# Pushing to Firebase (Create)

push() creates a new key (a “push ID” encoding timestamp and random bits) and returns a Firebase reference to it, so we can call setValue() on it.

```
public void add(MovieQuote movieQuote) {
    //DONE: Remove the next line(s) and use Firebase instead
    mMovieQuotesRef.push().setValue(movieQuote)
}
```

Test this and see the backend change:

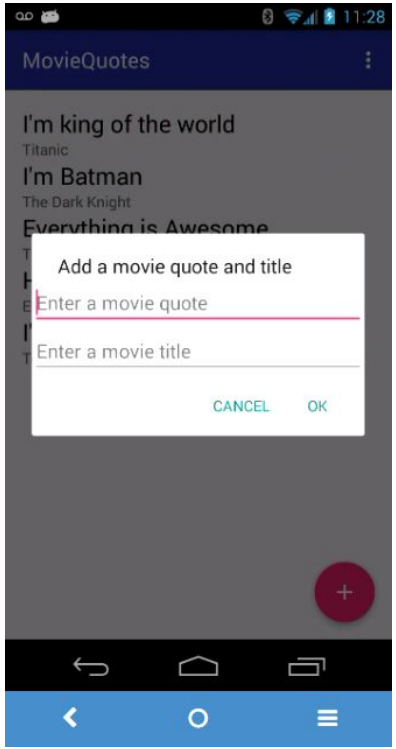



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Firebase EventListeners

In this lesson we'll learn how to listen for data added to Firebase.




ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# We don’t add the quote locally … yet

It doesn’t yet have a key assigned by Firebase.

So if we try to delete it, we’ll have issues.

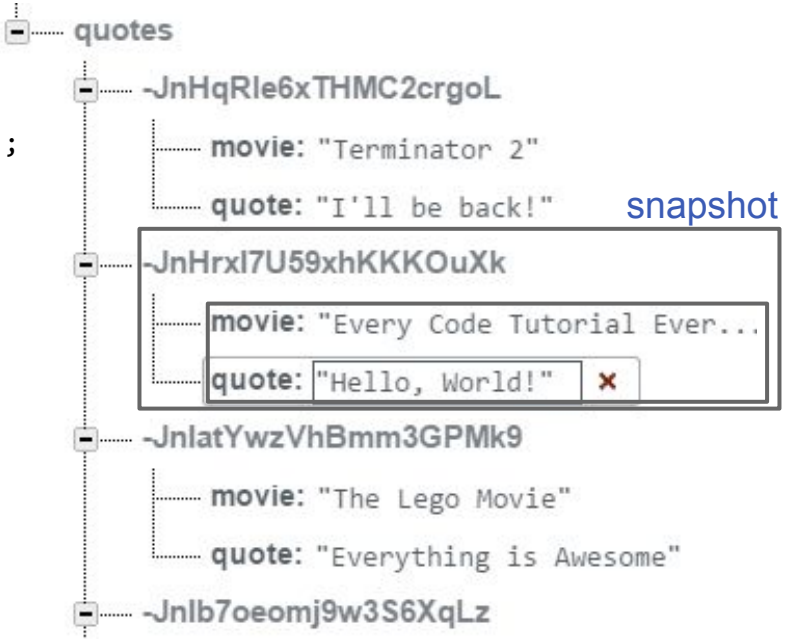
We’ll listen for the addition from Firebase and add the quote then.

Don’t worry, this will all happen almost instantaneously!

# We want to be notified when new quotes are added

Quotes are **children** of the quotes path. And to listen to events on children, add a **Firebase ChildEventListener.**

```
mMovieQuotesRef =
FirebaseDatabase.getInstance().getReference().child("quotes");
```



# Set a ChildEventListener on your Firebase object.

Declaring a nested class that implements **ChildEventListener** is a nice pattern.

```
    mMovieQuotesRef = new Firebase(QUOTES_PATH);
    mMovieQuotesRef.addChildEventListener(new QuotesChildEventListener());
}

private class QuotesChildEventListener implements ChildEventListener {

    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String s) {

    }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String s) {

    }

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {

    }

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String s) {

    }

    @Override
    public void onCancelled(FirebaseError firebaseError) {

    }
}
```

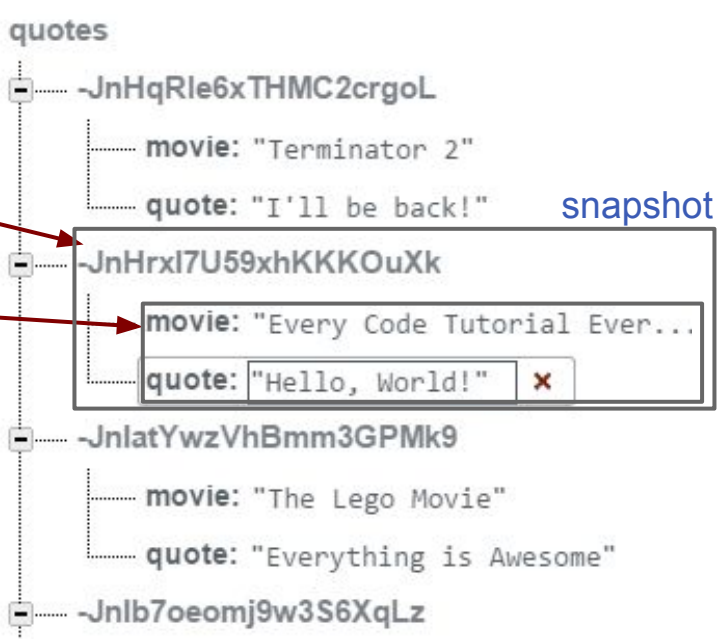
# All of the callbacks for the child events contain a dataSnapshot

We'll use these next

```
mMovieQuotesRef =
FirebaseDatabase.getInstance().getReference().child("quotes");
```

dataSnapshot.getKey();

dataSnapshot.getValue();



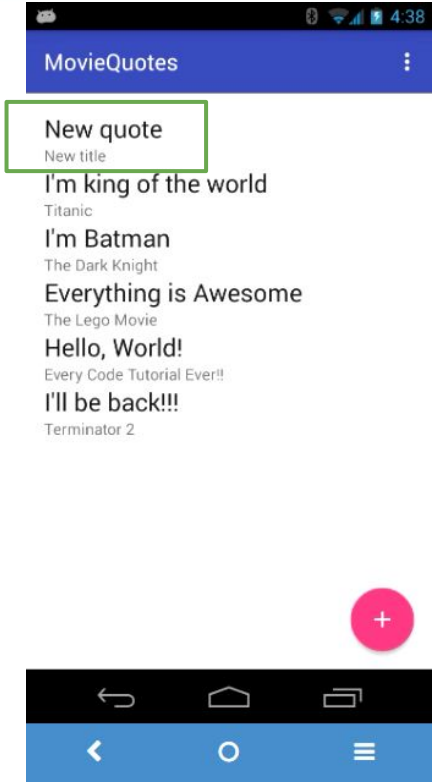
# onChildAdded (Read)

```java
@Override
public void onChildAdded(DataSnapshot dataSnapshot, String s) {
    // Passing a class to getValue tells it what class to deserialize the JSON to.
    MovieQuote quote = dataSnapshot.getValue(MovieQuote.class);
    // We set the key ourselves.
    quote.setKey(dataSnapshot.getKey());
    // Add it to our local list and display it.
    mMovieQuotes.add(0, quote);
    notifyDataSetChanged();
}
```

Called when a new child is added under the Firebase reference, in our case this is our new MovieQuotes under the quotes path of our repo.

# Run it!

If you run your app, you should get all quotes. Including ones you add.

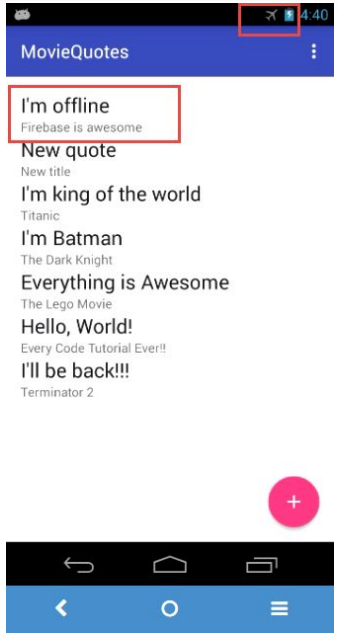


# For speed, it calls the listener even before making the round trip

This means:

1. We see changes immediately **even if we are offline.**(To see this, put your device in airplane mode and add an item.)

2. What if someone was editing the same item while offline?
   The latest change clobbers the others.
   - Ignores times of actual changes
   - Edit to a deleted item brings it back



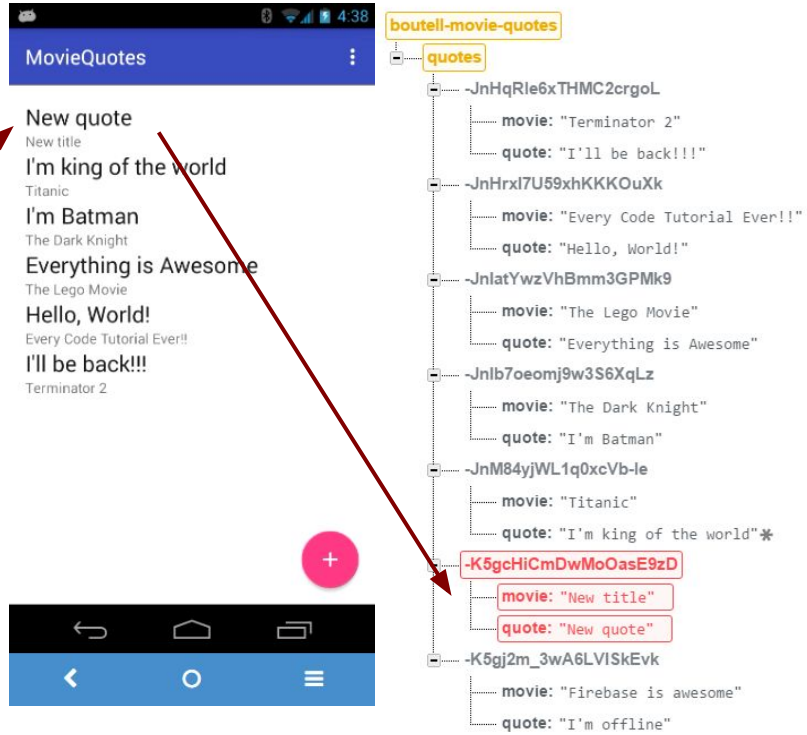


https://firebase.google.com/docs/database/android/offline-capabilities

# Remove

In this lesson we'll remove data from the Firebase and listen for that deletion.

Long press




ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Remove

Firebase references have a removeValue() method.

Which reference? The key of the quote to remove. And keys are children of the quote reference.

```
public void remove(MovieQuote movieQuote) {
    //DONE: Remove the next line(s) and use Firebase instead
    mMovieQuotesRef.child(movieQuote.getKey()).removeValue();
}
```

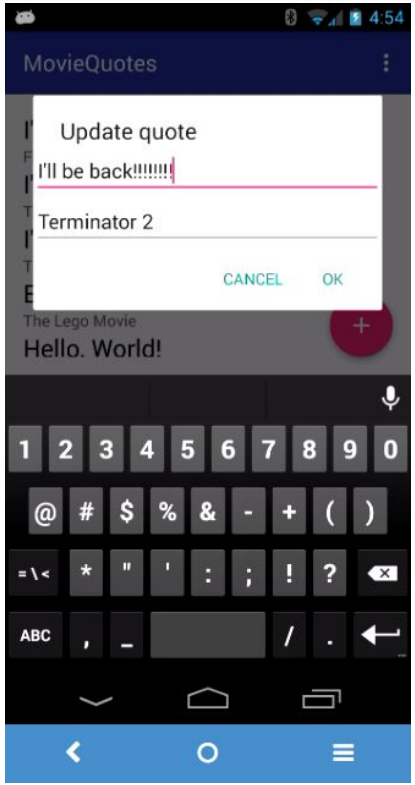
# onChildRemoved (Delete)

Called when a child under the Firebase reference is deleted. Test this.

```
@Override
public void onChildRemoved(DataSnapshot dataSnapshot) {
    String key = dataSnapshot.getKey();
    // Remove the item with the given key.
    for (MovieQuote mq : mMovieQuotes) {
        if (mq.getKey().equals(key)) {
            mMovieQuotes.remove(mq);
            break;
        }
    }
    notifyDataSetChanged();
}
```

# Update

In this lesson we'll update data in the Firebase and listen for that change.

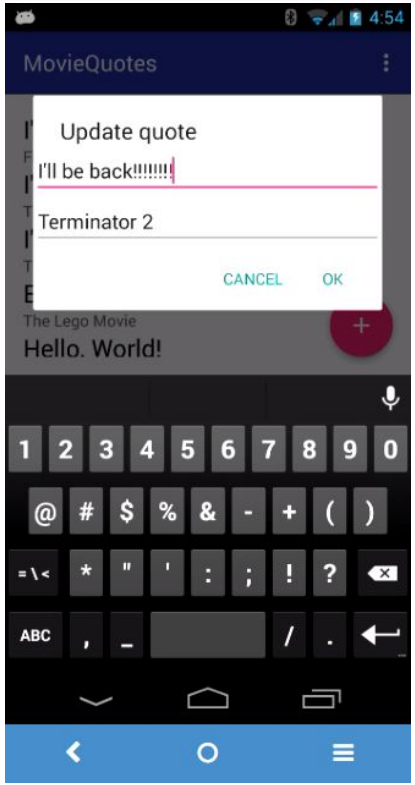



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Editing Data in Firebase (Update)

Instead of pushing a new key to the list, we are simply going to set the values at an existing element in the list

```
public void update(MovieQuote movieQuote, String newQuote, String newMovie) {
    //DONE: Remove the next line(s) and use Firebase instead
    movieQuote.setQuote(newQuote);
    movieQuote.setMovie(newMovie);
    mMovieQuotesRef.child(movieQuote.getKey()).setValue(movieQuote);
}
```

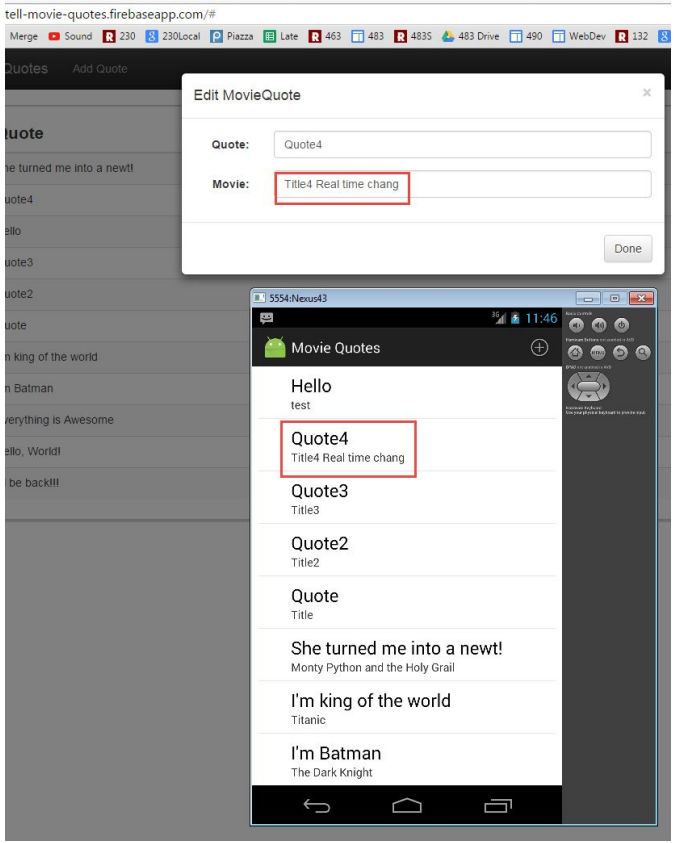


# onChildChanged (Update)

Called when a child under the Firebase reference is modified.

```
@Override
public void onChildChanged(DataSnapshot dataSnapshot, String s) {
    String key = dataSnapshot.getKey();
    MovieQuote updatedMovieQuote = dataSnapshot.getValue(MovieQuote.class);
    for (MovieQuote mq : mMovieQuotes) {
        if (mq.getKey().equals(key)) {
            mq.setValues(updatedMovieQuote);
            notifyDataSetChanged();
            return;
        }
    }
}
```

# Real time changes when you edit on the website




ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Finishing up the ChildEventListener

Empty since we aren't moving items around.

```
@Override
public void onChildMoved(DataSnapshot dataSnapshot, String s) {
    // empty
}
```

Called when the caller doesn't have permission to read the database.

Best practice is to include this.

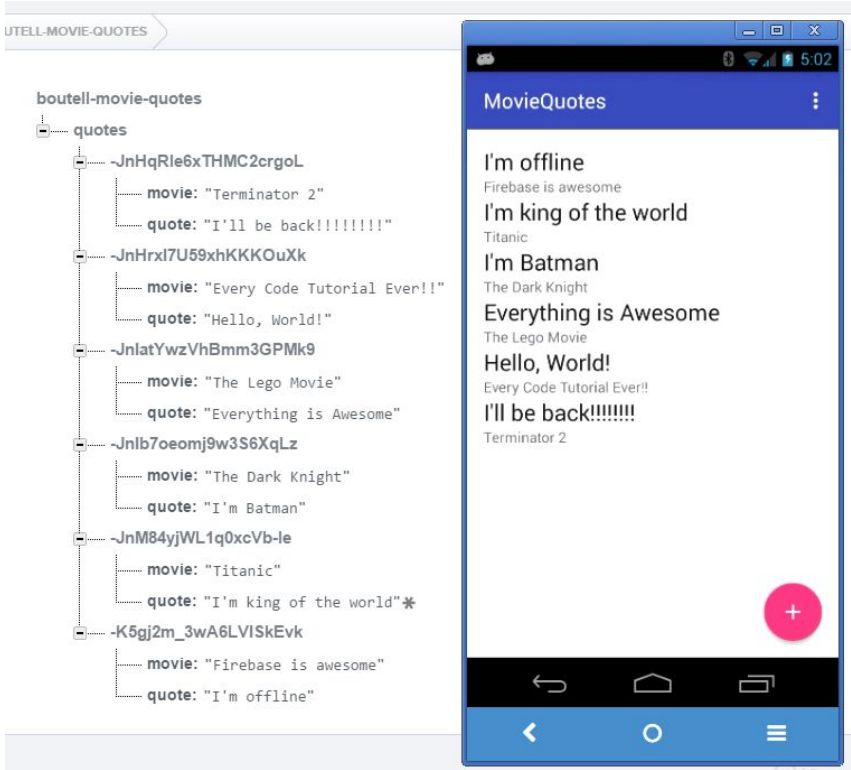
```
@Override
public void onCancelled(FirebaseError firebaseError) {
    Log.e("MQ", firebaseError.getMessage());
}
```

# Summary: The device in completely in sync with Firebase

Our Android client makes changes to the backend, viewable in the console and web client.

And our Android client gets changes from the console and the web client!

Feel free to post a friendly quote to a classmate's web client




ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# One more note about offline persistence

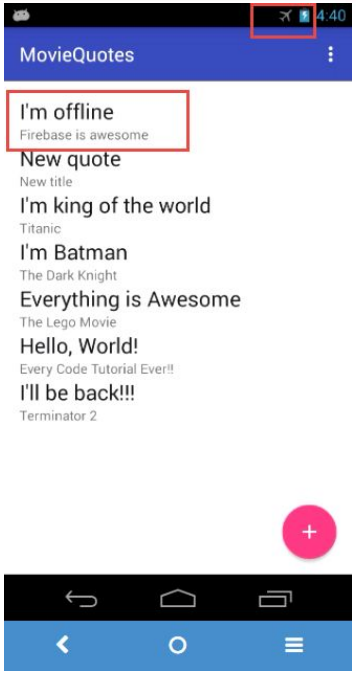
Data is persisted while offline, but only while the app is running. BUT, you can add true persistence with only two function calls:

On startup, add:

```
FirebaseDatabase.getInstance().setPersistenceEnabled(true);
```

And for every Firebase path you want to persist, add the reference to the path's root:

```
mMovieQuotesRef.keepSynced(true);
```



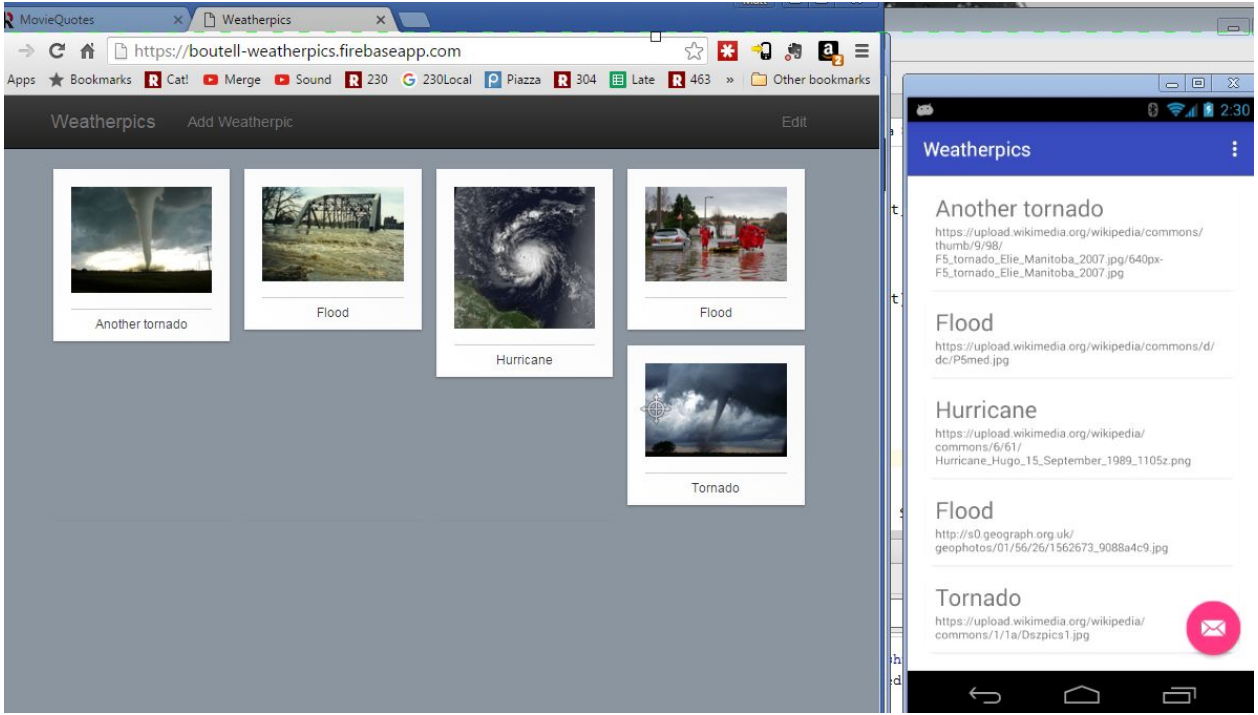

Goodbye, SQlite?

https://firebase.google.com/docs/database/android/offline-capabilities

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# Firebase lab: PhotoBucket (formerly Weatherpics)

- Practice with Firebase pushes and listeners
- Downloading photos
- Fragments

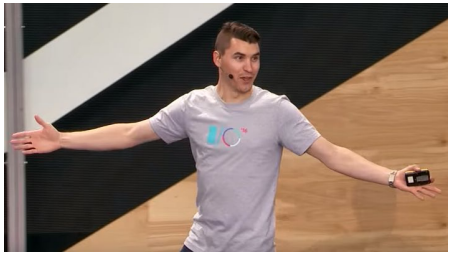



ROSE-HULMAN INSTITUTE OF TECHNOLOGY

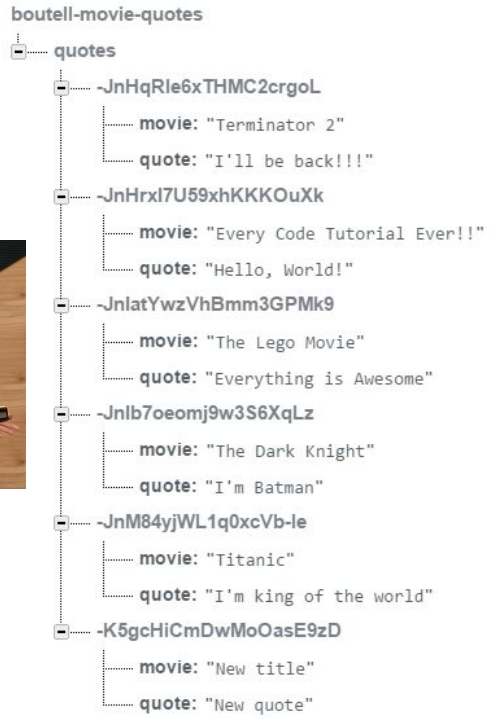
# Summary: Firebase gives easy synchronization!

They have SDKs for Web and iOS clients too, you could create a whole platform from Firebase!

Live demo at I/O ‘16:
from zero to app in 34 min.



If you have questions then refer to their in-depth documentation.




ROSE-HULMAN INSTITUTE OF TECHNOLOGY

# More Rose connections at Firebase

Mike McDonald ‘14 CpE/CS

Engineer @ Google (Firebase)

@asciimike, mpmcdonald@

Mike took CSSE483.

He returned to introduce us to Firebase.

Now Firebase is integral to our workflow.

What will **you** do in the next 5 years?

Alex Memering CS ‘15 and
Tyler Rockwood, CS/SE ‘16 are also there

ROSE-HULMAN INSTITUTE OF TECHNOLOGY