Google Firebase and Android

Google Firebase: a bit of history

- Firebase Real-time Database was created in 2011 as an API to synchronize application data across iOS, Android, and Web devices
- Data was stored on Firebase's cloud
- Google acquired Firebase in 2014
- Since then, Google Firebase has been offering several services, critical for development of mobile and other software systems

Google Firebase: services

- Google Firebase includes:
  - Firebase Authentication
  - Firebase Realtime Database
  - Cloud Firestore (follow up to Realtime DB)
  - Firebase Storage
  - Firebase Hosting
  - Firebase Cloud Messaging
  - ML Kit
  - Google Analytics

Firebase Realtime Database

Firebase Realtime Database

- It is a NoSQL (Non-relational or non-SQL) database
- Can be used as a service to store data and share data from mobile apps
- Web applications can use it too
- Data is stored as a JSON object, usually with many nested objects and arrays

NoSQL databases

- Databases that store data in other forms than relational tables
- Examples include:
  - Key-value stores (dbm, BerkeleyDB, Redis)
  - Document stores (Firebase, MongoDB, DocumentDB, CounchDB, BaseX)
  - Graph databases (Neo4j, TigerGraph, AllegroGraph)
  - RDF stores (Virtuoso, Amazon Neptune, Jena)
  - Object databases (GemStone, ObjectStore)

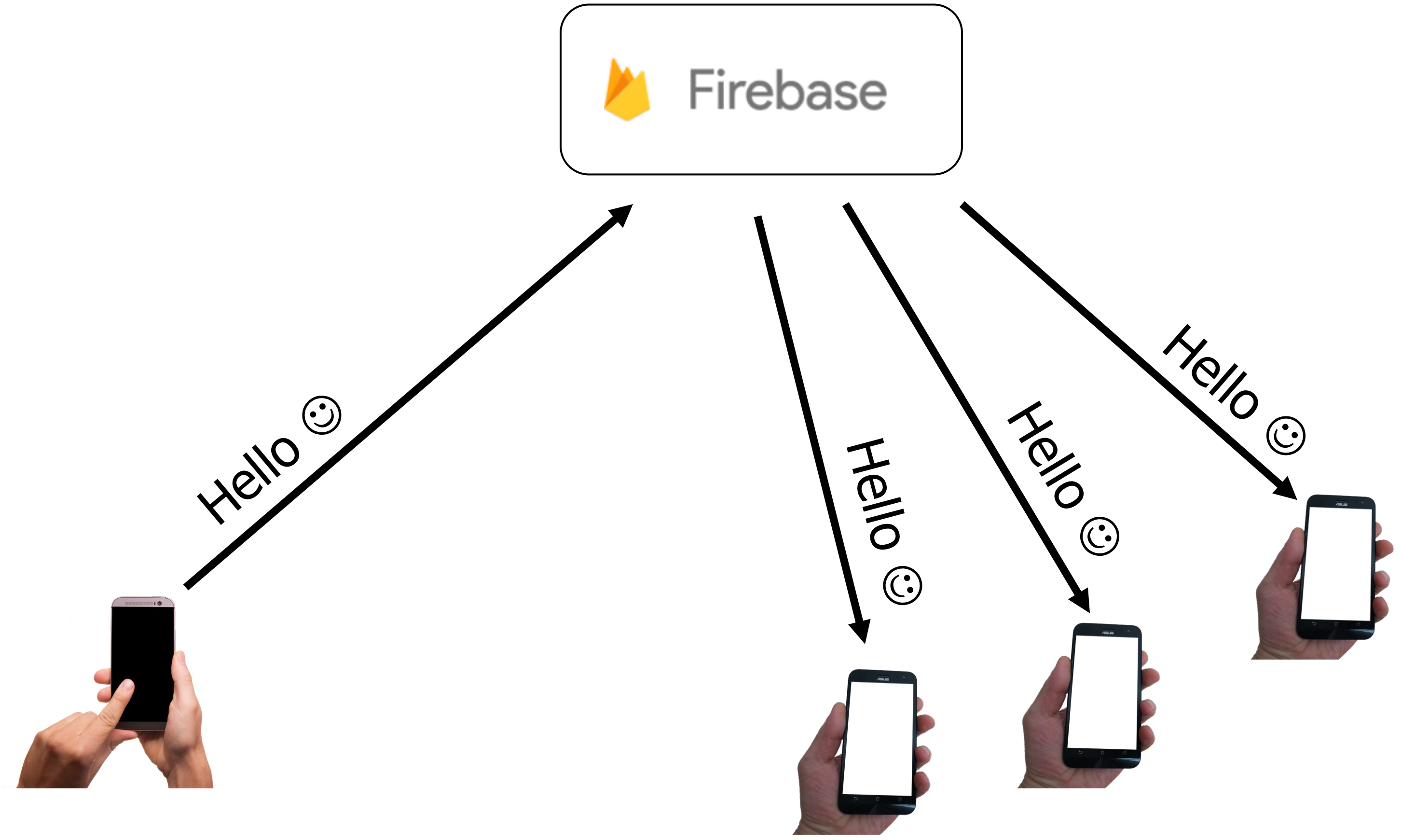
Firebase Cloud Firestore

- Cloud Firestore is Firebase's newest database for mobile app development
- It is a successor to the Realtime Database and offers a different, perhaps more intuitive data model
- Cloud Firestore features more powerful, faster queries and is more scalable than the Realtime Database
- However, we will not use it in class this semester (but you may use it in your final project)

Firebase Realtime Database

- Back to the Realtime Database...
- Data can be synchronized across many clients, in nearly *real time*
- For example, one mobile client can write/update data in Firebase and thousands of clients listening for changes are notified (get the new data) in real time – virtually instantly
- Firebase can support thousands of clients

Firebase Realtime Database



Some data, e.g., a message, can be made available to *thousands* of interested clients in real time, *virtually instantly*

Firebase Realtime Database

- Firebase users can be added and authenticated
- The user information is kept separate from the actual data
- Android app may rely on the Firebase API to read/write data to a Realtime Database
- RESTful interface is also available:

  https://firebase.google.com/docs/reference/rest/database

Firebase Setup

- Setup Android Studio for Firebase

  https://firebase.google.com/docs/android/setup

  I strongly suggest using the Firebase console (Option 1)

- Basic steps to use Firebase are:
  - Create a project in the Firebase console
  - Add the Firebase SDK to the app
  - Configure database rules (authentication)
  - Implement reading/writing in the app

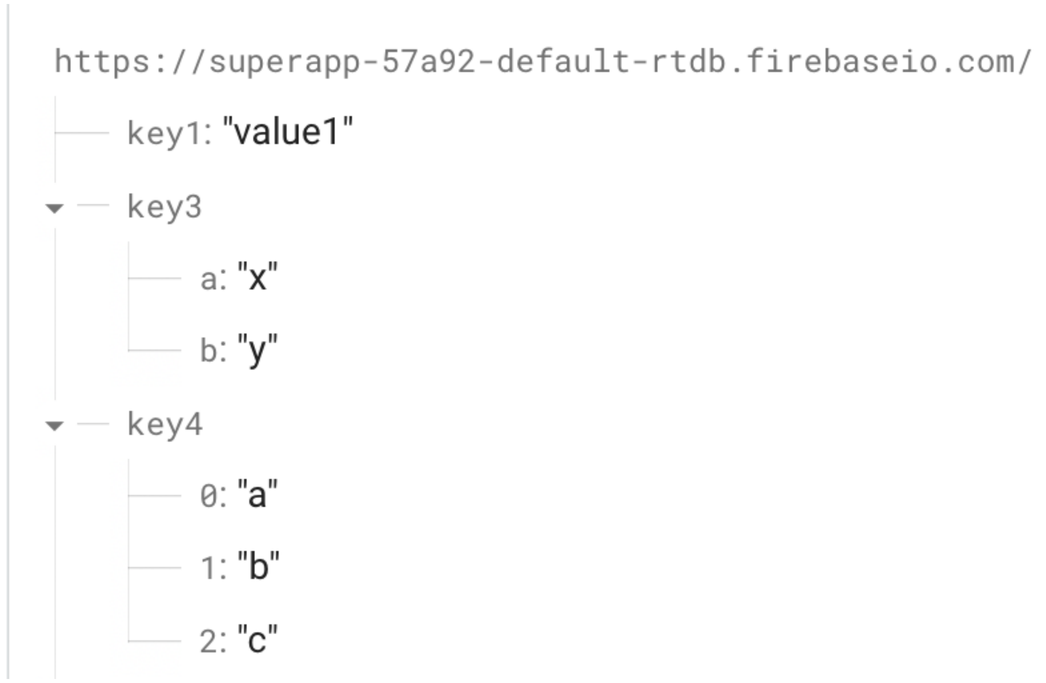
Firebase Documentation

- You should read Google documentation on integrating your Android app with Google Firebase:

firebase.google.com/docs/android/setup?authuser=0

Firebase Documentation

- Firebase Realtime Database is a JSON database, and all data is stored as a *single JSON object*
- Nested objects or arrays may be used



```
https://superapp-57a92-default-rtdb.firebaseio.com/
  key1: "value1"
▾ key3
    a: "x"
    b: "y"
▾ key4
    0: "a"
    1: "b"
    2: "c"
```

Firebase Authentication

- Managing Firebase users

  https://firebase.google.com/docs/auth/android/manage-users

- Creating a user
  - A new user can be created using the `createUserWithEmailAndPassword()` method
  - A user can sign in using a federated identity provider, such as Google Sign-In or Facebook Login, as well.
  - We will use email/password here in our app

Firebase Authentication

- Fragment of code to create a new user

```
String email = emailEditText.getText().toString();
String password = passworEditText.getText().toString();

FirebaseAuth mAuth = FirebaseAuth.getInstance();

mAuth.createUserWithEmailAndPassword( email, password )
    .addOnCompleteListener( RegisterActivity.this, new
         OnCompleteListener<AuthResult>() {
              @Override
              public void onComplete(Task<AuthResult> task) {
                   if (task.isSuccessful()) {
                        Toast.makeText( getApplicationContext(),
                                     "Registered user: " + email,
                                     Toast.LENGTH_SHORT ).show();
```

Firebase Authentication

- Fragment of code to login, for example in a LoginButton listener:

```
List<AuthUI.IdpConfig> providers = Arrays.asList(
                new AuthUI.IdpConfig.EmailBuilder().build()
        );

startActivityForResult(
                AuthUI.getInstance()
                        .createSignInIntentBuilder()
                        .setAvailableProviders(providers)
                        .setIsSmartLockEnabled(false)
                        .build(),
                RC_SIGN_IN);
```

Firebase Authentication

- Must create a listener to get the started activity’s result:

```
protected void onActivityResult( int requestCode, int
        resultCode, Intent data) {
    super.onActivityResult( requestCode, resultCode, data );
    if( requestCode == RC_SIGN_IN ) {
        IdpResponse response =
                IdpResponse.fromResultIntent( data );
        if( resultCode == RESULT_OK ) {
            FirebaseUser user =
                FirebaseAuth.getInstance().getCurrentUser();

            Log.i( "Test", "Signed in as: " + user.getEmail() );
```

Firebase Realtime Database

- Intro to Reading and writing

  https://firebase.google.com/docs/database/android/read-and-write

- Writing to a database

```
// Write a message to the database

FirebaseDatabase database =
                    FirebaseDatabase.getInstance();

DatabaseReference myRef =
                    database.getReference( "message” );

myRef.setValue( "Hello :)” );
```

Firebase Realtime Database

- The `setValue` method (on a database reference) can be used with:
  - String
  - Long
  - Double
  - Boolean
  - Map<String, Object>
  - List<Object>

Firebase Realtime Database

- Reading from a database

```
// Read from the database value for ”message”

DatabaseReference myRef = database.getReference( "message” );

myRef.addValueEventListener( new ValueEventListener() {
    @Override
    public void onDataChange( DataSnapshot dataSnapshot ) {
        // This method is called once with the initial value and again
        // whenever data at this location is updated.
        String value = dataSnapshot.getValue( String.class );
    }
    @Override
    public void onCancelled( DatabaseError error ) {
        // Failed to read value
        Log.d( TAG, "Failed to read value.", error.toException() );
    }
});
```

Firebase Realtime Database

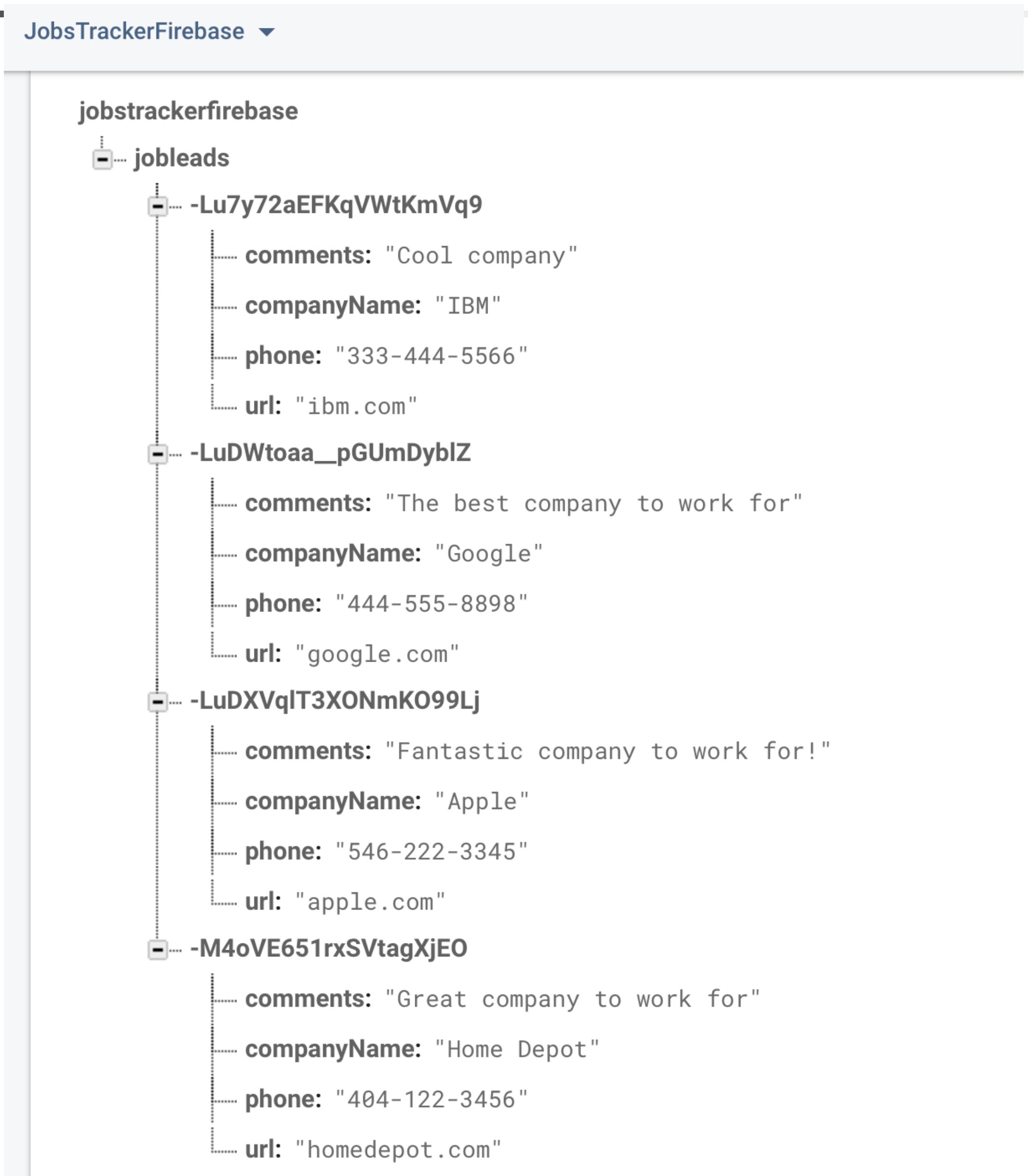
- Working with lists

  https://firebase.google.com/docs/database/android/lists-of-data

  - To append a new element, use `push()` on a list reference; it returns a unique reference for the new list element
  - You can use `setValue` to set the new element's value
  - Unique list element values are based on timestamps and automatically ordered chronologically

Firebase Realtime Database

Example list shown in Firebase console



```
JobsTrackerFirebase ▾

jobstrackerfirebase
  jobleads
    -Lu7y72aEFKqVWtKmVq9
        comments: "Cool company"
        companyName: "IBM"
        phone: "333-444-5566"
        url: "ibm.com"
    -LuDWtoaa__pGUmDyblZ
        comments: "The best company to work for"
        companyName: "Google"
        phone: "444-555-8898"
        url: "google.com"
    -LuDXVqIT3XONmKO99Lj
        comments: "Fantastic company to work for!"
        companyName: "Apple"
        phone: "546-222-3345"
        url: "apple.com"
    -M4oVE651rxSVtagXjEO
        comments: "Great company to work for"
        companyName: "Home Depot"
        phone: "404-122-3456"
        url: "homedepot.com"
```

Firebase Realtime Database

- You can attach listeners (callbacks) to a list reference to observe changes in list elements:
  - `onChildAdded()`
  - `onChildChanged()`
  - `onChildRemoved()`
  - `onChildMoved()`
- You can retrieve the whole list by attaching a `ValueEventListener`

JobsTracker Using Firebase

- We will modify our JobsTracker app, once again
- This time, we will use the Firebase Realtime Database instead of an SQLite database or a REST service
- A JobsTracker app using Firebase will be available on eLC, in the Sample apps folder

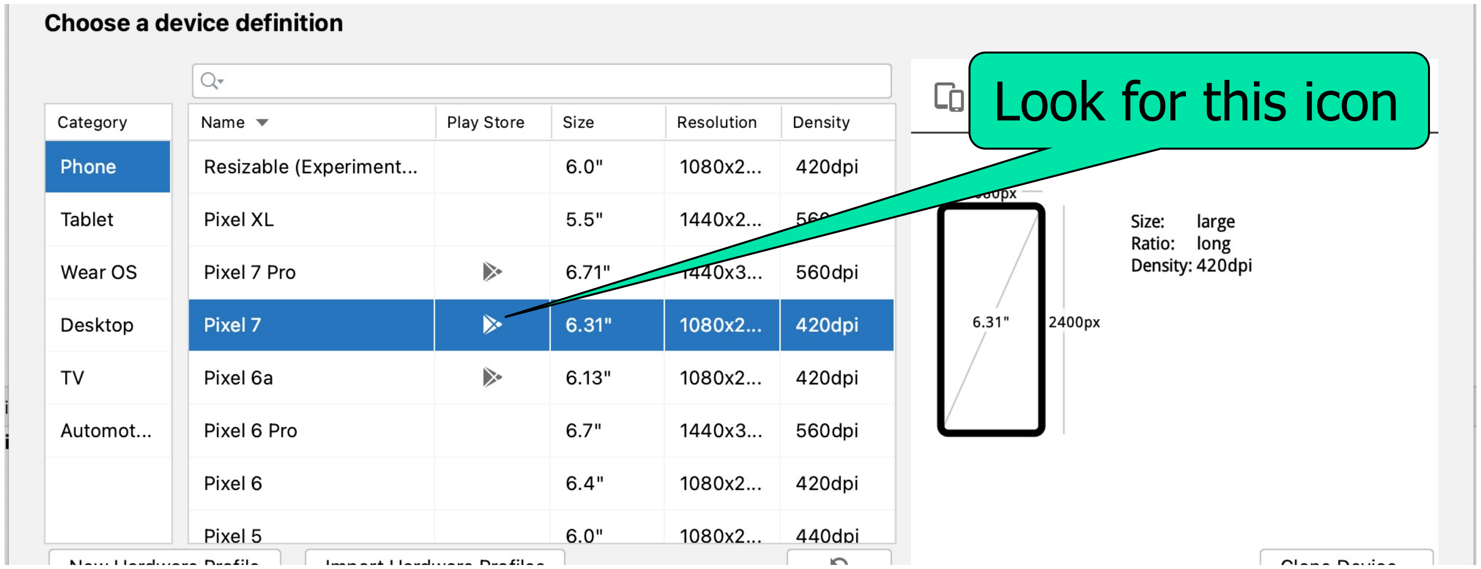
JobsTracker: a Firebase project

- You will need to create your own Firebase project (with Realtime Database) for the JobsTracker app (using similar steps, as shown in the lecture)
- You will need to provide a correct package name in the project
- Download the `google-services.json` file and replace the *placeholder* file `google-services.json`, which is included in the app on eLC; use the Project view (not Android)

JobsTracker: a Firebase project

- IMPORTANT: A recent change at Firebase requires you to use an AVD with Play Store installed!
- As a result, your current AVD will likely not work with Firebase
- So, create a new AVD with a Play Store pre-installed
- Hint: the icon for the Play Store must be visible, when you pick a new AVD

JobsTracker: a Firebase project



Choose a device definition

Category	Name	Play Store	Size	Resolution	Density
Phone	Resizable (Experiment...		6.0"	1080x2...	420dpi
Tablet	Pixel XL		5.5"	1440x2...	560dpi
Wear OS	Pixel 7 Pro	▶	6.71"	1440x3...	560dpi
Desktop	Pixel 7	▶	6.31"	1080x2...	420dpi
TV	Pixel 6a	▶	6.13"	1080x2...	420dpi
Automot...	Pixel 6 Pro		6.7"	1440x3...	560dpi
	Pixel 6		6.4"	1080x2...	420dpi
	Pixel 5		6.0"	1080x2...	440dpi

Size: large
Ratio: long
Density: 420dpi

- A good choice is a Pixel 7, which is already predefined in Android Studio; it has a good “skin” available
- The API 30-31 is a good Android version for this new AVD