Identifying the files changes

* By reading the details of unique commits of the fork. We were able to identify which file was changed.

Categorized these file as java, Manifest File layout, other resources file, gradle or others

To identify if the file is a java, we look at the extension. Java

By looking at the file name AndroidManifest.xml to identify manifest files

By looking at the file path if it contains the word layout to identify layout file.

Other resource files such as images, color, strings, styles. Are all calorized under other resources by looking at the file path containing res directory but does not contain layout directory.

Looking at the java file

* We further categorize these java file as Activity, Fragments, Service, Broadcast Receiver, Content Provider, others.
* If java file extends from a parent class which as the word Activity. The we call it activity. (And also looking at the manifest file if it has been called under activity tag).
* If the extends from parent class containing the word Service, then we call it Service. (And also looking at the manifest file if it has been called under service tag).
* If the extends from parent class containing the word Receiver, then we call it Broadcast Receiver (And also looking at the manifest file if it has been called under receiver tag).
* If the extends from parent class containing the word Fragment, then we call it Fragment.
* To identify the content provider, We looked at both the imports and or parent class or the usage if it contains atleast one of this word:

- firebasestorage- Cloud Storage is built for app developers who need to store and serve user-generated content, such as photos or videos.

Cloud Storage for Firebase is a powerful, simple, and cost-effective object storage service built for Google scale. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality. You can use our SDKs to store images, audio, video, or other user-generated content. On the server, you can use [Google Cloud Storage](https://cloud.google.com/storage), to access the same files.

- storage

- Observable: The Data Binding Library allows you to make objects, fields, or collections observable. Any plain-old object can be used for data binding, but modifying the

- Cursor : This interface provides random read-write access to the result set returned by a database query.

Cursor implementations are not required to be synchronized so code using a Cursor from multiple threads should perform its own synchronization when using the Cursor. Implementations should subclass [AbstractCursor](https://developer.android.com/reference/android/database/AbstractCursor.html)

- database : Contains classes to explore data returned through a content provider.

If you need to manage data in a private database, use the [android.database.sqlite](https://developer.android.com/reference/android/database/sqlite/package-summary.html) classes. These classes are used to manage the [Cursor](https://developer.android.com/reference/android/database/Cursor.html) object returned from a content provider query. Databases are usually created and opened with [openOrCreateDatabase(String, int, SQLiteDatabase.CursorFactory)](https://developer.android.com/reference/android/content/Context.html#openOrCreateDatabase(java.lang.String,%20int,%20android.database.sqlite.SQLiteDatabase.CursorFactory)) To make requests through content providers, you can use the [content.ContentResolver](https://developer.android.com/reference/android/content/ContentResolver.html) class.

All databases are stored on the device in /data/data/<package\_name>/databases

- contentvalues : This class is used to store a set of values that the [ContentResolver](https://developer.android.com/reference/android/content/ContentResolver.html) can process

- execsql : Compiles an SQL statement into a reusable pre-compiled statement object. The parameters are identical to execSQL(String) . You may put ?s in the statement ...

- provider: A content provider is a subclass of ContentProvider that supplies structured access to data managed by the application. All content providers in your application ...

- ContentObserver: Receives call backs for changes to content. Must be implemented by objects which are added to a [ContentObservable](https://developer.android.com/reference/android/database/ContentObservable.html).

-preferences: A node in a hierarchical collection of preference data. This class allows applications to store and retrieve user and system preference and configuration data.

- session: Android Profiler continues to collect profiling data until you disconnect the device or click End Session. Figure 1. Android Profiler shared timeline view. 1 Android

- sql : Provides the API for accessing and processing data stored in a data source (usually a relational database) using the JavaTM programming language. This API includes a framework whereby different drivers can be installed dynamically to access different data sources. Although the JDBCTM API is mainly geared to passing SQL statements to a database, it provides for reading and writing data from any data source with a tabular format. The reader/writer facility, available through the javax.sql.RowSet group of interfaces, can be customized to use and update data from a spread sheet, flat file, or any other tabular data source

- DataClip: Representation of a clipped data on the clipboard.

ClipData is a complex type containing one or more Item instances, each of which can hold one or more representations of an item of data. For display to the user, it also has a label.

- ContentResolver: This class provides applications access to the content model.

We then categories each java file category as added, Removed, Modified and Renamed. By looking at the status returned from the Github api.