## Android Fundamentals Project Self-Evaluation

**Instructions:** Once you’ve completed your Final Project, please respond to the questions below. This is a chance for you to briefly explain to the grader your thought-process during development. Once you are done, include this with the source code and accompanying files you are submitting. Then, give yourself a pat on the back for making a great app!

# Questions about Required Components

## Permissions

**Please elaborate on why you chose the permissions in your app.**

|  |
| --- |
| The permissions my app needs are in two categories below :-  <!-- Permissions required for HashTrace to access the network and perform network acess -->  <uses-permission android:name=*"android.permission.INTERNET"* /> - application needs to initiate network communication by opening network sockets  <uses-permission android:name=*"android.permission.ACCESS\_NETWORK\_STATE"* /> -receive information about the network  <uses-permission android:name=*"android.permission.ACCESS\_WIFI\_STATE"* /> - for users who will use my app in a proxy network environment, I need to check their wifi state  <!-- Permissions required by the sync adapter -->  <uses-permission android:name=*"android.permission.READ\_SYNC\_SETTINGS"* /> - will allowus to read the sync settings  <uses-permission android:name=*"android.permission.WRITE\_SYNC\_SETTINGS"* /> - will allow application to write syncronisation settings  <uses-permission android:name=*"android.permission.AUTHENTICATE\_ACCOUNTS"* /> - Will allow my application to act as an AccountAuthenticator for the AccountManager  <uses-permission android:name=*"android.permission.WRITE\_EXTERNAL\_STORAGE"*/> - Allows application to write to external storage |

## Content Provider

**What is the name of your Content Provider, and how is it backed? (For example, Sunshine’s Content Provider is named WeatherProvider backed by an SQLite database, with two tables: weather and location.)**

|  |
| --- |
| The name of my ContentProvider is TweetHashTracerContentProvider and it is backed by an SQLite database with three tables -:   * Hashtag – for storing hashtag string * tweet and – for storing the tweets * tweetfavour – for storing favourited yweets over time |

**What backend does it talk to? (For example, Sunshine talks to the OpenWeatherMap API.)**

|  |
| --- |
| My app communicates with twitter API and depends on it for retrieval of tweets. |

**If your app uses a SyncAdapter, what is it called? What mechanism is used to actually talk over the network? (For example, Sunshine uses HttpURLConnection to talk to the network, but your app may use a third-party library to do the talking.)**

|  |
| --- |
| Yes, my app uses a SyncAdapter called TweetHashTracerSyncAdapter. This app uses the twitter4j library to access the twitter API through HttpURLConnection . |

**What loaders/adapters are used?**

|  |
| --- |
| App has a   * TweetList Adapter which extends a CursorAdapter for building the tweet list view. * I also use a SwipePagerAdapter which extends a SmartFragmentStatePagerAdapter for building my fragments. * The TweetListFragment implements LoaderCallBacks providing it with a loader to retrieve database data. |

## User/App State

**Please elaborate on how/where your app correctly preserves and restores user or app state. (See rubric for examples on this question)**

|  |
| --- |
| The app correctly preserves and restores user or app state. For example :-   * Activity state is preserved even if a rotation or change in orientation occurs, this is because it uses loaders which are not tied entirely to activity’s state. * Selected item on a list view is maintained even upon a rotation of screen or change in orientation * User text input is also preserved for instance when entering the #HashTag value, the value entered persists even on rotation. * When the app is resumed after the device wakes from sleep (locked) state, the app returns the user to the exact state in which it was last used. * When the app is relaunched from Home or All Apps, the app restores the app state as closely as possible to the previous state. |

# Questions about Optional Components

Answer the questions that are applicable to your final project

## Notifications

**Please elaborate on how/where you implemented Notifications in your app:**

|  |
| --- |
| Notifications for new tweets is implemented, but is limited to tweets below a certain date span from the current time, hence notifications will only show if such tweets exist. Tweets are batched and come as one notification. |

## ShareActionProvider

**Please elaborate on how/where you implemented ShareActionProvider:**

|  |
| --- |
|  |

## Broadcast Events

**Please elaborate on how/where you implemented Broadcast Events:**

|  |
| --- |
|  |

## Custom Views

**Please elaborate on how/where you implemented Custom Views:**

|  |
| --- |
| I have used custom views such as the :-   * PullUpToRefreshListView for the TweetListFragment . It is a special ListView for a refresh functionality. * LineChartView for the GraphFragment to display the graphs, it is part of the HelloCharts library . |